DTIC FILE COPY

AD

AD-E402 131

Technical Report ARFSD-TR-90021

AUTOMATED ACQUISITION OF COPPERHEAD STOCKPILE SURVEILLANCE TEST DATA

AD-A231 387

John P. Tobak

December 1990

U.S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER

US ARMY
ARMAMENT MUNITIONS
& CHEMICAL COMMAND
ARMAMENT RDE CENTER

Fire Support Armaments Center

Picatinny Arsenal, New Jersey

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER Technical Report ARFSD-TR-90021	5. MONITORING ORGANIZATION REPORT NUMBER
6a. NAME OF PERFORMING ORGANIZATION ARDEC, FSAC — 6b. OFFICE SYMBOL SMCAR-FSP-S	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (CITY, STATE, AND ZIP CODE) Precision Munitions Division Picatinny Arsenal, NJ 07806-5000	7b. ADDRESS (CITY, STATE, AND ZIP CODE)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION ARDEC, IMD STINFO Br — 8b. OFFICE SYMBOL SMCAR-IMI-I	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (CITY, STATE, AND ZIP CODE) Picatinny Arsenal, NJ 07806-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (INCLUDE SECURITY CLASSIFICATION)

AUTOMATED ACQUISITION OF COPPERHEAD STOCKPILE SURVEILLANCE TEST DATA

12. PERSONAL AUTHOR(S)

John P. Tobak

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (YEAR, MONTH, DAY)	15. PAGE COUNT
Progress	FROM Jun 89 TO Jun 90	December 1990	67

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (CONTINUE ON REVERSE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)		
FIELD	GROUP	SUB-GROUP	Basic	Test system	Data acquisition
			Instrumentation	Test set calibration	Automatic data acquisition
			Copperhead	Computer aided test	Test set

19. ABSTRACT (CONTINUE ON REVERSE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

This report provides a description of the software and related hardware used to acquire test data for the Copperhead stockpile surveillance program. It outlines their interaction and provides an example of how the test system is used. Methods for system calibration are also presented as well as a scheme for sampling data in bursts. Selected portions of in-house developed software are also discussed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (INCLUDE AREA CODE)	22c. OFFICE SYMBOL
I. HAZENDARI	(201) 724-3316	SMCAR-IMI-I

DD FORM 1473, 84 MAR

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

INTRODUCTION

The Copperhead stockpile surveillance program presently being developed by the Advanced Technologies Laboratory, ARDEC, is a highly automated software driven test system. It is designed to test the Copperhead round's seeker, electronics package, guidance, and control systems. The data obtained will be used to identify trends in round performance and subsequently provide a means of predicting future round reliability.

This report describes the software and related hardware used to acquire test data for the Copperhead stockpile surveillance program. It outlines their interaction and provides an example of how the test system is used. Methods for system calibration are also presented as well as a scheme for sampling data in bursts. Selected portions of in-house developed software are also discussed.

GENERAL SYSTEM DESCRIPTION

Software

The test software used to support the Copperhead stockpile surveillance effort consists of three programs developed in-house and two packages commercially available through Hewlett Packard. The first of the three in-house programs is named COPPER and acts to perform the actual testing of a Copperhead unit. While COPPER is able to plot and list test results, a second program, PLOT_OL, is available for off-line visual data scrutiny. Data are also able to be tested against specifications using the third available program, ANALYZE. All of the in-house programs are written in Hewlett Packard (H.P.) BASIC which is a cross between FORTRAN and traditional BASIC.

The first of H.P.'s packages is called FTM/300. It acts as an executive through which the in-house programs are executed. FTM/300 serves as an umbrella under which programs have access to user created tables designed to organize, enhance, simplify, and coordinate equipment direction. The second H.P. product is named CAT (computer aided test). It is used by COPPER as a library of routines from which to program and drive the family of cards which comprise H.P.'s 6944 multiprogrammer instrumentation package.

Hardware

The 6944 multiprogrammer instrumentation package is a computer-aided test system which provides the engineer with a versatile means of test system development. Its mainframe allows the insertion or deletion of a variety of test system component

cards able to be accessed on a dedicated data bus by means of an H.P. 320 personal computer. Examples of the cards available are analog-to-digital converters, relay scanners, counters, timers, digital-to-analog converters, and relay cards.

While the 6944 multiprogrammer is a large contributor of hardware to the system, it is not alone and is supplemented by a host of instrumentation programmable by means of the IEEE-488 data bus. This bus is an industry standard and is independent of the 6944. It is therefore able to support the equipment of manufacturers other then Hewlett Packard; however, H.P. equipment has been used in an attempt to achieve maximum system continuity. Examples of the equipment used over the bus include power supplies, function generators, counters, digital voltmeters, attenuators, stepper motors, plotters, and printers. A third bus is also used to provide high speed data transfer between the H.P. 320 and data storage equipment. H.P. 9133 and 7958 hard disk drives are presently available on the system in addition to an H.P. 9144 tape backup unit.

WINDOWED DATA COLLECTION

Hardware

The H.P. 6944 multiprogrammer handles most of Copperhead's stockpile surveillance testing by means of a multiwindow/multiline testing environment. This is done to allow data sampling to be halted temporarily during a test while system modifications are made to accommodate changes in test requirements. Modifications might include changes in the data lines being sampled, the number of lines looked at simultaneously, or the rate at which they are being sampled. Other changes could include a variation in system stimulus or a modification in system loading. A description of the environment's operation can be obtained from figure 1. This diagram depicts a scanner/pacer produced clock pulse (Pout) which triggers the simultaneous sampling of data lines by A/D converters and the data's subsequent storage into memory. The same pulse train also defines the duration of an initial data collection window by means of a count limit programmed into the counter. Upon reaching the count limit, a pulse is generated at the counter's Carry-not output which stops the scanner/pacer and starts the timer simultaneously. The counter is reprogrammed at this time to a count limit representative of the next data collection window's length. The scanner/pacer remains idle until the completion of an exact and predefined idle time measured by the timer. The scanner/pacer is then restarted by the timer's completion of process (Cop-not) pulse and the second data collection window is begun. The process is repeated until three data collection windows have been realized.

Software

The COPPER test program begins by asking the user to specify tests which are to be executed from the menu of figure 2. Upon doing so, the user marks certain driver subroutines for execution. Each of these routines has been written and customized specifically for the test it represents. Once testing begins, each driver routine calls upon a central testing routine called Test_frame (app A) which acts to implement the windowing system previously mentioned. Scanner channel assignments, window lengths, sampling rates, and scanning schemes are all activated with this routine. While sampling data within the system's data collection windows, driver routine defined options are implemented which allow actions to be taken as data is being collected. Such actions are implemented in the Utility_switch subroutine of appendix A and may include the switching of relays, attenuator changes, system input modifications, metering, or positional changes in Copperhead itself. Once a particular test has been completed, it passes control to the next driver routine marked for execution and the process continues.

Test Example

The following is an example of one of the 30 tests which presently use the windowed data collection method discussed in this report. The test is called Squib_driv and it makes use of all three of the system's data collection windows. The Squib_driv routine (app A) is no more than a half a page in executable code length. Inspection of the routine provides the user with immediate information regarding the sampling rate of each data collection window, the length of each, and the scanner channel assignments during each window. The sampling rates of each of the three windows are defined by variables Time_per_chan_1, Time_per_chan_2, and Time_per_chan_3. Variables Window_1, Window_2, and Window_3 specify window lengths. Finally, the scanner channels used can be obtained from the values of the W1, W2, and W3 prefixed variables listed. Variables W1L1 and W1L2 for example specify the first and second data lines of window one as scanner channels 0 and 32. It should be mentioned at this point that only two data lines can be sampled in the first and third data collection windows. Window two will accommodate four data lines of simultaneously sampled data. The names of each of the eight data lines monitored in this test are specified by the name$ variables. They can be seen to have been defined as EA1 SQ DR, EA2 SQ DR, 30V BAT, SGG SQ DR, GAS SQ DR, SGS SQ DR, WUN SQ DR, and WEX SQ DR. Once defined, all variables are used during Squib driv's calling of the Test_frame routine and the test's subsequent execution. Output plots of each of the three data collection windows are represented in figures 3 through 5.

CALIBRATION

Data Interface

The circuit of figure 1 depicts one of the four data interfaces used to acquire Copperhead test data. Each of the interfaces consists of a voltage divider network, relay scanner, buffer amplifier, and analog to digital (A/D) converter. The buffer amplifier acts as an interconnection between the data line being monitored and the A/D converter. It provides a low output impedance required by the A/D and a high input impedance to the data line. The relay scanner offers an economical and manageable means of multiple data line monitoring while the voltage dividers reduce Copperhead outputs to voltage levels unable to saturate the buffer amplifier. The interfaces are used in concert to monitor up to four data lines simultaneously or as many as 128 in a multiplexed fashion. Sampling rates of 2 microseconds per sample can be realized along with a total data collection limit of 60,000 data points per test. Measurement accuracies are limited to a value of 0.05% due to the resolution of the A/D range being used.

Method of Calibration

To obtain accuracies as high as 0.05%, it is necessary that the divider gains along with the gains and offsets of the buffer-amp/A-to-D systems be precisely measured. A calibration is performed for this purpose providing gain and offset values which are then stored and later used to calculate values for Vin from recorded values of Vo.

Copperhead calibration begins with the buffer-amp/A-to-D systems. Voltages from -10 V to +10V are supplied to each of the systems without the divider loads attached. A/D output voltages are obtained for each value of the input voltages applied. A linear curve is fit to the data using a least squares method. The curve's resulting slope is representative of the system's voltage gain while its intercept is equivalent to the system offset voltage. Such a calibration makes it possible to adjust for changes in gain and offset while distributing errors due to variations in the system's linearity. The gain and offset obtained from the calibration of a particular buffer-amp/A-to-D system are next used to determine the gains associated with each of the dividers used on that system.

The development of an equation for the divider's gain begins with an expression for A/D output voltage, Vo, from the circuit of figure 1.

$$Vo = Dg^{*}Sg^{*}Vin + So \qquad (1)$$

Variable Dg represents the gain of the divider network while Sg and So symbolize the buffer-amp/A-to-D system's gain and offset respectively. The divider's gain is obtained directly from equation 1 in the form of equation 2.

$$Dg = (Vo - So)/(Sg*Vin) \quad (2)$$

The accuracy of Dg is directly related to the value of Vin at which equation 2 is evaluated. This is seen from figure 6's demonstration of how small instrument error in the measurement of Vo cause significant errors in divider gain when calibrating at small values of Vin. As a result, equation 3's subsequent determination of Vin (also depicted by fig. 6) produces large voltages errors when large values of Vo are collected. In other words, percentage error experienced during calibration is the percentage error to be expected during application.

$$Vin = (Vo - Sc)/(Sg*Dg) \quad (3)$$

Therefore, Dg's calibration should be done at as high an input voltage value as possible in an attempt to minimize errors associated with instrument accuracy and resolution. This is accomplished through the application of a high voltage pulse narrow enough so as not to exceed divider power limitations.

The calibration method just described produces measurement accuracies as high as 0.05% while eliminating the need for a time consuming -10 V to +10 V calibration for each voltage divider.

Calibration Software

Data interface calibration is initiated by the user's test selection menu (fig. 2) choice of Ad_ld_cal. Ad_ld_cal is found in the appendix and begins by assigning test operation to the second data collection window. This is accomplished by making the value of (Window_#/Time_per_chanl_#) greater than one for # = 2. Window_2 represents the window's length in seconds while Time_per_chanl_2 specifies the time between data samples. A window length and sampling time of 63 seconds and 15 ms respectively can be seen to have been specified.

Variables Sw_windows and Sw_state are also defined in Ad_ld cal for use in later subroutines. Once in Test_frame, Sw_windows is used to direct program operation to subroutine Utility_switch where Sw_state then leads the logic to subroutine Drv_ps1c_rd_dvm. It is in this routine that voltages are applied to the inputs of the four data interfaces via scanner channels not involving divided inputs. Input voltages are stepped from -10 V to +10 V in 1 0 V increments and read through the use of a digital voltmeter. Output voltage measurements are collected by the A/D converters between the time the scanner is started and stopped as passes are made through the routine's FOR NEXT loop. After collection of the data is complete, control passes to the Plot_channels subroutine where the data is used to generate linear curve fit parameters representative of the voltage gains and offsets associated with each data interface. These parameters are saved and made immediate use of during the second phase of data interface calibration.

The second phase of data interface calibration takes place completely within Ad_ld_cal. Upon the program's return from Test_frame, each scanner channel is driven by a 40 ms wide pulse of as high a voltage level as possible. Again, this is done to obtain high calibration accuracy without exceeding divider power limitations. Input DVM and output A/D voltages are again collected but this time they are used along with the curve fit parameters of the first phase to determine the exact value of each channel's divider gain. This is accomplished through the use of equation 2. An example of the portion over which a typical input waveform is sampled is given in figure 7. Because of the inability of the input source to supply a perfectly flat pulse, care must be taken during this phase of the calibration to assure that input and output samples are simultaneously collected. Once the data have been properly collected and calculated, results are stored along with fit data for subsequent use in equation 3.

Gaussian Input

Many of the tests performed on a Copperhead unit require the input of a Gaussian type pulse used to simulate the laser light seen by the round during operation. An H.P. 8112A pulse generator and H.P. 8496 attenuator are used to supply this pulse to a Copperhead unit during testing. The Gaussian pulse setup used with the exception of the DVM and the reconnection of channels 1, 2, 4, and 5 as Copperhead inputs are depicted in figure 8. Since one generator is able to supply a pair of attenuators, only two are needed to generate the four Gaussian pulses required. Because the amplitude of each pulse must be supplied exactly according to the specification, a calibration of the pulse generator/attenuator combinations must be performed prior to testing.

Method of Calibration

Calibration of the Gaussian input pulse begins with the determination of the pulse's amplitude. While the test's specifications provide pulse generator output and attenuation values normally sufficient for test execution, they assume the use of a pulse generator with an output amplitude capability above that of the HP 8112. As a result, it is necessary to determine the actual Copperhead input pulse amplitude required so that new specifications can be obtained for pulse generator amplitudes and attenuator settings. This is accomplished through the use of equation 4's definition of dB.

$$dB = 20 * \log_{10}(Vcop/Vps) \tag{4}$$

Variable Vcop of equation 4 represents the amplitude of the signal to be seen by Copperhead while Vps is the pulse generator amplitude required by the specification. Solving for Vcop and specifying the attenuation required as a positive number, yields the following:

$$Vcop = Vps/(10^{(dB/20)}) \tag{5}$$

A new value for dB can now be obtained based on an empirically chosen pulse generator amplitude of 5 V. These new values for pulse generator amplitude and attenuator dB are then implemented to produce a pulse amplitude which is checked against the desired value for Vcop. If necessary, fine adjustments are then made to the pulse generator and attenuators to bring the resulting pulse in line with the specification. Since attenuations as great as 100 dB down from 10 volts must be calibrated, the accuracy of a digital voltmeter is used to measure pulse amplitude (fig. 8). A DC measurement is made of the attenuators output over 170 cycles of a 1 kHz 50% duty cycle square wave input. The measurement is then doubled to obtain the output pulse height. A square wave input is used instead of a Gaussian because of the nonlinear relationship between DC measurements made on Gaussian curves of different amplitude. The calibration procedure just described is done entirely by the software in an automated fashion. It can be performed on a test by test basis by selecting the appropriate C prefixed test name from the test selection menu of figure 2.

Calibration Software

When one of the test agenda's C prefixed tests is executed, it calls a routine named Set_dB_out. This routine determines and stores pulse generator and attenuator settings which together realize a specified Gaussian pulse. The following is an example of such a call:

```
Set_dB_out(2,50,60,8)
```

In this example, attenuator group 2 (fig. 8) is set so that the output of attenuators C and D produce respective attenuations of 50 dB and 60 dB down from 10 volts. A test number of 8 identifies the calibration data during its subsequent calibration data storage. Inspection of subroutine Set_dB_out reveals the execution of equation 5 at line 10881 and its use to obtain a new dB value on line 10887. A measured value of true dB resulting from the implementation of new values of pulse generator output and attenuator settings is obtained on line 10910. Fine adjustments to the pulse generator and attenuators are then made followed by the storage of calibration data on line 10994. These values are later recalled and used a. pulse generator and attenuator settings during testing.

BURST SAMPLING

Most of the tests performed on a Copperhead unit do not demand extensive data storage. There are some tests however which monitor data lines over long periods of time at a high sampling rate. These tests would exceed the data storage limits of the system if sampling were performed in a continuous manner. Since continuous sampling is not a requirement during these tests, it is possible to sample data in a burst fashion. During a burst type test, measurements are taken at high sampling rates for short periods of time. This type of burst sampling is of particular value when the amplitude of an infrequently occurring narrow pulse is to be determined. It provides a means of acquiring the many samples needed to average and eliminate the effects of noise without exhausting available memory.

Burst Hardware

A schematic of the circuit used to implement burst sampling is shown in figure 9. It is a reconfiguration of the hardware of figure 1 and results in the realization of bursting at the expense of windowing. Switches Sw1, Sw2, and Sw3 are normally open for window data collection but closed while bursting. Switch Sw4 is open for bursting but closed during windowing. Switches 5 and 6 are both closed during windowing but only switch 6 becomes opened while bursting. Figure 9's operation is as follows: Once the Start signal goes and remains positive, AND gate A1's output is enabled to follow the Burst Trigger input. A1's output is counted at input 1 of the counter with every negative transition of Input 1. Concurrently, the output of invertor Inv is applied to the external trigger (Ext-not) of the scanner card. When enabled, this input triggers the card with every negative transition it sees. This in turn causes a burst of periodic A/D triggers by means of the scanner's Pout line. Initially however, the scanner's Ext-not input is purposely locked via software to introduce a delay before bursting begins. Once a software defined delay has been reached by Input 1, the Borrow-not output pulses low permanently unlocking the U1-not input by means of AND gate A2. This enables the scanner's Ext-not input and permits periodic pulses to be produced at Pout with every positive excursion of the Burst Trigger input. The counter's Eop-not line also goes low at this point causing Input 1 to be premanently disabled. Simultaneously to the unlocking of the scanner's Ext-not input, Ext-not of the counter is also activated causing another previously set value to be loaded into the counter's count register. This is the limit count which defines the length of a burst and is reached by input A's counting of Pout's pulses. Once this limit is reached, the Carry-not signal pulses low. This causes the Scanner to be turned off and the counter's count register to be reset to the burst limit by way of Ext-not. Sampling remains halted until the next positive edge of the Burst Trigger input. The timer card's only function is to provide an output which is triggered by the counter's Borrow-not signal upon completion of its initial delay.

Burst Software

As a result of the reassignment of system components from a windowing to a bursting environment, interruptions in the bursting period cannot be made without relinquishing the start of the test as a time reference. Such interruptions are therefore not permitted. Consequently, most of the software associated with bursting is pre-test related and can be found in the subroutine called Burst_setup. The routine begins at line 11024 by placing switches Sw1, Sw2, Sw3, and Sw4 in their respective burst mode conditions. The timer and counter are then cleared followed by the already discussed disabling of the scanner's Ext-not input. The counter is first set at line 11049 to a value representative of a delay before bursting. The duration of each burst is then set on line 11054. While the delay count is only processed once by the counter, the burst count is reloaded and used after every execution of the counter's Ext-not input. The test is then begun by the relay closure of line 11070. The next line waits for the completion of the sampling delay and then enables analog to digital converter number one to accept data. Data are collected until memory is locked out on line 11085. The scanner is then stopped and the system's configuration is returned to the windowing mode.

MAJOR SUBROUTINES

When one of the 30 driver routines which comprise Copperhead's stockpile surveillance program is executed, it calls upon other routines to perform many of the operations which are common to all of the tests. Three such subroutines are Test_frame, Utility_switch, and Store_run_data. While many other routines are also used during testing, these three represent the heart of the software and provide an overall picture of the testing process.

Test_frame

The Test_frame routine appears in the appendix and begins with line 300's setting of the four system A/D ranges. Ranges from 0.1 volts to 100 volts are available. Calculations are then made starting at line 410 to obtain the number of triggers to be counted by the counter during each window of data collection. The routine continues on line 770 by setting the system's counter and timer to values representative of the test's first data collection window and idle time respectively. Memory pointer settings follow on line 920 along with the specification of scanner channel assignments and sampling period.

Three options are available at the point of test initiation (line 1230). Tests can begin from within the burst subroutine, by the software start of line 1370 or by way of the relay closure of line 1410. Once begun, the test can either move into the Utility_switch subroutine or continue to the looping code of statement label Spin1. Program execution loops at this point until a message is received from the counter announcing the completion of the first data collection window. The second data collec-

tion window's system parameters are then set followed by another wait loop (label Spin2) which monitors the condition of the timer and its indication that Idle_time_1 is complete. Upon Idle_time_1's completion, the timer is reset to the second idle time while testing enters the second data collection window. Again the test is offered the option of entering Utility_switch or waiting for the completion of the window. The third data window is treated in the same fashion as the first and second. Upon its completion, the test data is up loaded to the computer via the Input_rblock statement of line 3410. The data is then stored onto hard disk via the Store_run_data subroutine. If the test is being run in the debug mode, the user has the option at this point of either listing or plotting the test data. When scrutinization of the data is complete, logic is directed to the next test to be executed.

Utility_Switch

Each of Test_frame's data collection windows provides an opportunity for the execution of subroutine Utility_switch. This routine's main function is to offer the user the ability to make system changes while sampling is in progress. Examination of the subroutine (app) demonstrates its use of the Sw_state variable. This variable was previously set in the test's driver routine and functions as a pointer to a particular course of action. Such actions include the setting of attenuations, pulse generators, and D/A converters; the calling of subroutines; or the opening and closing of relays. Utility_switch offers the user a great deal of flexibility and can also be called directly from the test's driver routine.

Store_run_data

The Store_run_data routine is also called during the execution of Test_fame. It acts to store down loaded 6944 test data onto hard disk. Data are taken from the 6944 in 20,000 sample groups generally representing the data collected during one data collection window. System restrictions then cause each 20,000 group to be broken up and finally stored in five 4,000 sample long records. The data are later retrieved for inspection by the Get_run_data routine.

ANALYZE

Most of the discussion thus far has dealt with the Copperhead stockpile surveillance program's data collection software. The collection of data however would be meaningless without a procedure for its Analysis. Data analysis is realized through the execution of the computer program called analyze. Its format is similar to that of the Copper program in that it consists of driver subroutines which are chosen from a menu and then used to drive one central testing or in this case, analyzing subprogram. Analyze's central analyzing subroutine is called Main_frame and is used to scrutinize the data collected on a data line by data line basis. Pulse heights and their times of

occurrence are check against test specifications and then reported along with determinations with regard to unit acceptance or failure. Analyze's Squib_driv routine (app) can be used as an example of the format used when data line levels, their edges of definition, and measurement tolerances are supplied to the program. Lines 11179 through 11197 depict matrix like arrangements of input data and their respective read statements. The columns of each matrix represent the data associated with one of the eight data lines able to be monitored during a test while each row represents another line condition to be met. During a Squib_driv analysis for example pulses of 10.77 volts $\pm$5 mv are checked for on lines one and two. Rising and falling edges of 5 ms and 25 ms $\pm$5 ms, respectively, must also be found if the test's specification is to be satisfied. Voltage levels are determined by an averaging of the data between a specified start time after level transition and the line's next transition in level. A typical level measurement is demonstrated in figure 10.

CONCLUSIONS

The testing methods and software used to support the Copperhead stockpile surveillance program have proven to be both accurate and versatile. This has been exemplified by their ability to readily support the different requirements of over 30 tests thus far. Windowed data collection has been seen to be an excellent means of accommodating a variety of test configurations within a standardized framework of operation. The many options available during a standard windowed test have provided the ability to customize the system to a particular test with a minimal of programming. The calibration methods used have produced the highest system accuracies possible and will, as a result, produce reliable Copperhead trend analysis data. The test system described in this report provides the Advanced Technologies Laboratory, ARDEC, with a ready means of collecting and analyzing data in an accurate and responsive manner. While the software written thus far satisfies the requirements necessary for the completion of Copperhead's electronic package testing, work on the round's live sequence test continues.

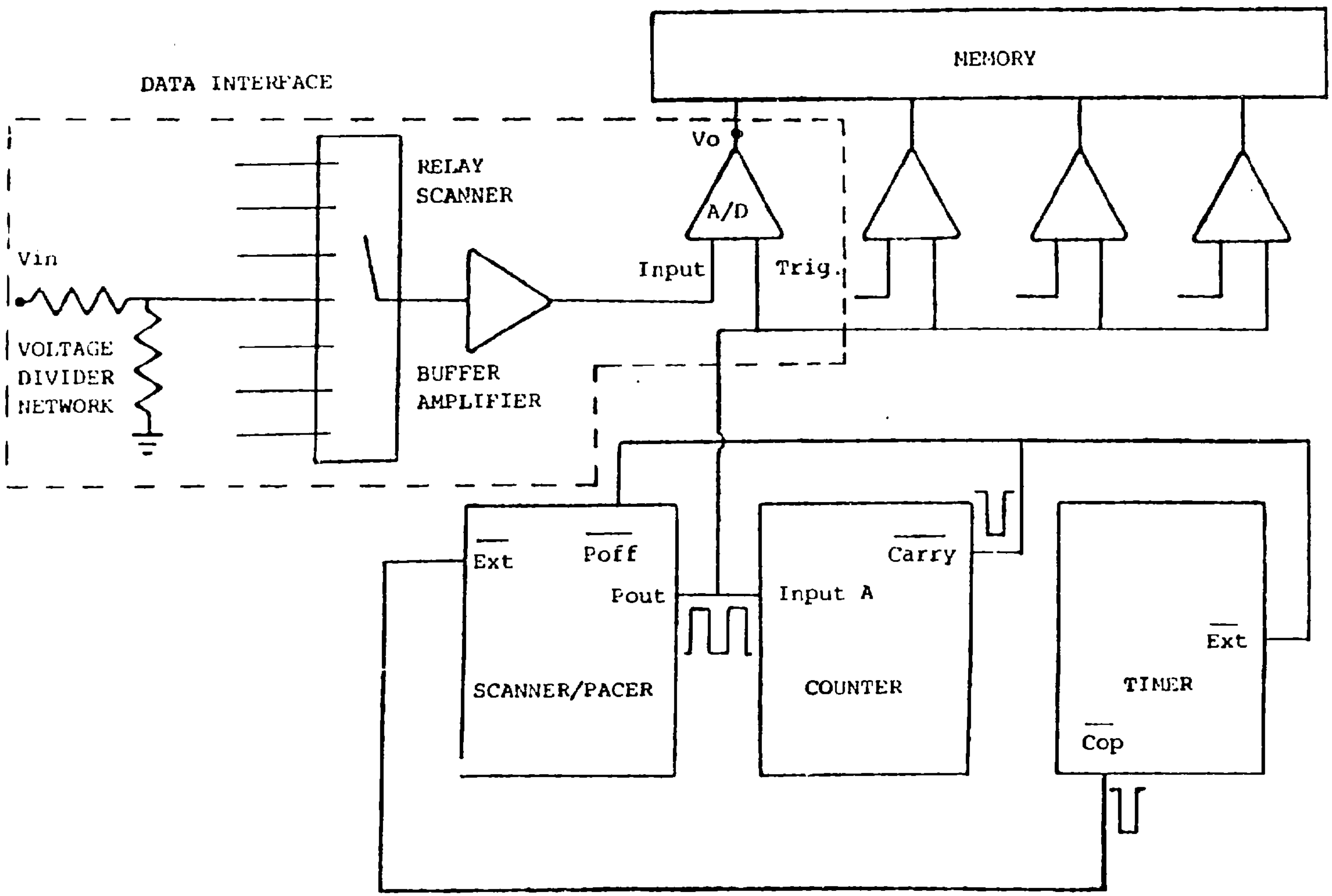


Figure 1. Widowed data collection hardware

TEST SELECTION MENU

SPECIFY A TEST AGENDA FROM THE FOLLOWING LIST USING THE ARROW KEYS AND THE COMMAND MENU BELOW. ONLY 'CONTINUE' AFTER ALL OF THE TESTS TO BE INCLUDED HAVE BEEN SELECTED AND MARKED BY AN '*'.

TA-GE-CDC	CAL	NOISE AGC	TST	SPIN TORQ	TST
NOISE-AGC-A	CAL	PWD	TST	FREE GYRO	TST
NOISE-AGC-B	CAL	SJI	TST	CAGE GAIN	TST
PWD	CAL	DYN THRESH	TST	GIM_CNSTRT	TST
SJI-A	CAL	CODING	TST	PCH_YAW_LS	TST
SJI-B	CAL	PULSE AGC	TST	GRAV_BIAS	TST
DYN-THRESH-A	CAL	SKR GMBL ANG	TST	GUID_GAIN	TST
DYN-THRESH-B	CAL	OPT COM PWR	TST	ATT_HLD_GN	TST
CODING	CAL	ROLL GAIN	TST	C_PYL_20	CAL
PULSE-AGC-A	CAL	A/D LOAD	CAL	C_PYL_30	CAL
PULSE-AGC-B	CAL	SWS DIS TST	TST	C_PYL_40	CAL
SQUIB DRIVE	TST	CONTINUITY	TST	C_PYL_50	CAL
TA-GE-CDC	TST	SPIN DRIVE	TST	C_PYL_60	CAL

CONTINUE (UN)SELECT NEXT PAGE SELECT ALL UNSELECT ALL

NOTE: This is page 1 of a 2 page menu.

Figure 2. Test selection menu

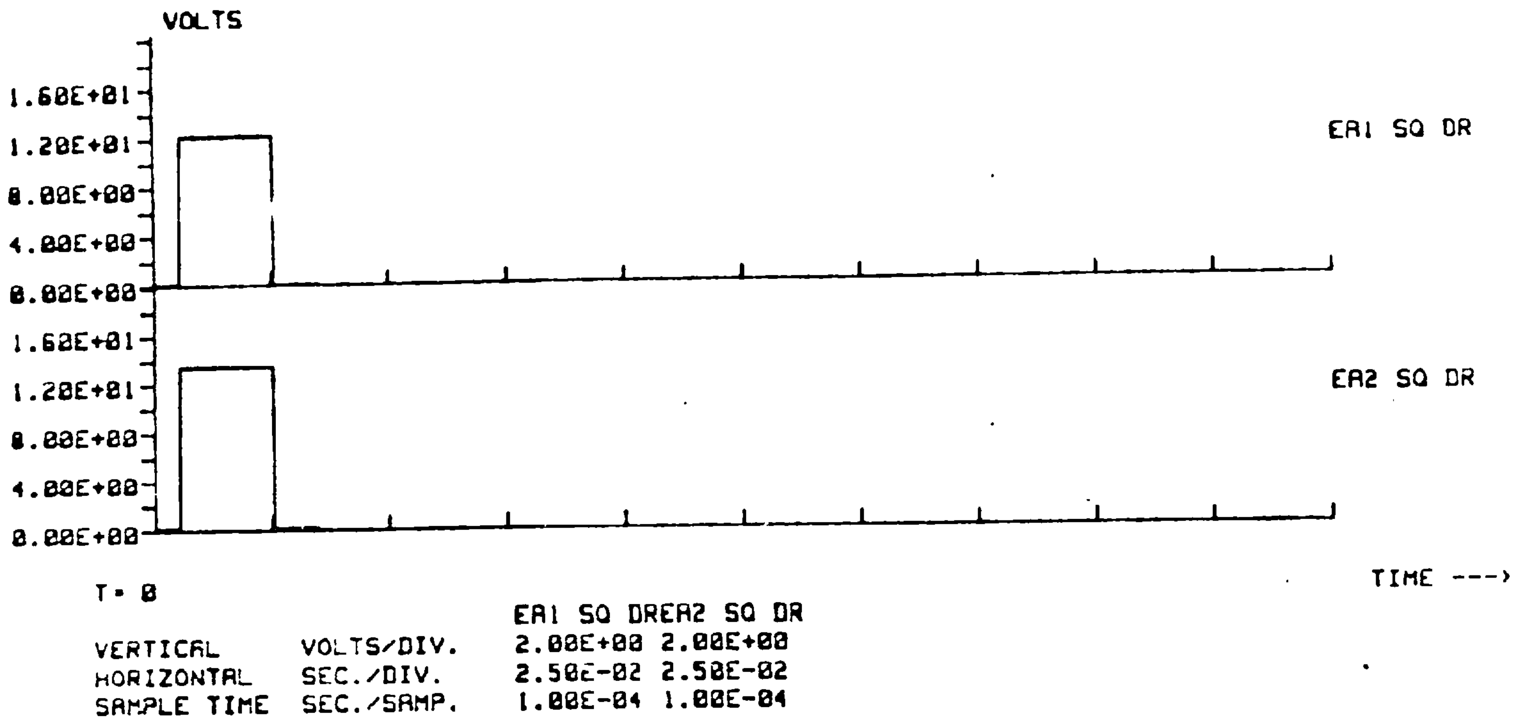
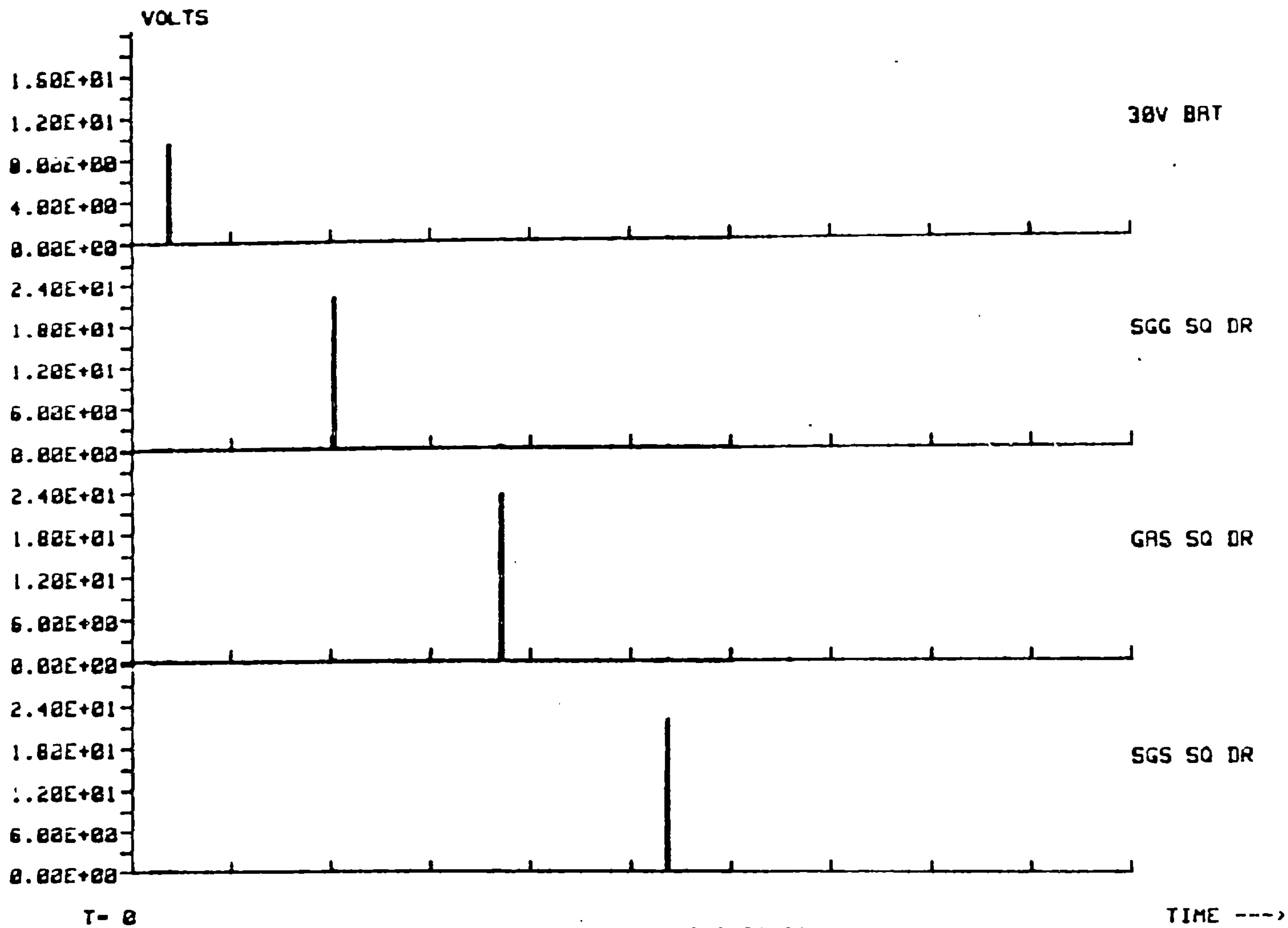


Figure 3. Window 1 Squib_driv data



		30V BAT	SGG SQ DR	GAS SQ DR	SGS SQ DR
VERTICAL	VOLTS/DIV.	2.00E+00	3.00E+00	3.00E+00	3.00E+00
HORIZONTAL	SEC./DIV.	6.00E-01	6.00E-01	6.00E-01	6.00E-01
SAMPLE TIME	SEC./SAMP.	2.00E-03	2.00E-03	2.00E-03	2.00E-03

Figure 4. Window 2 Squib_driv data

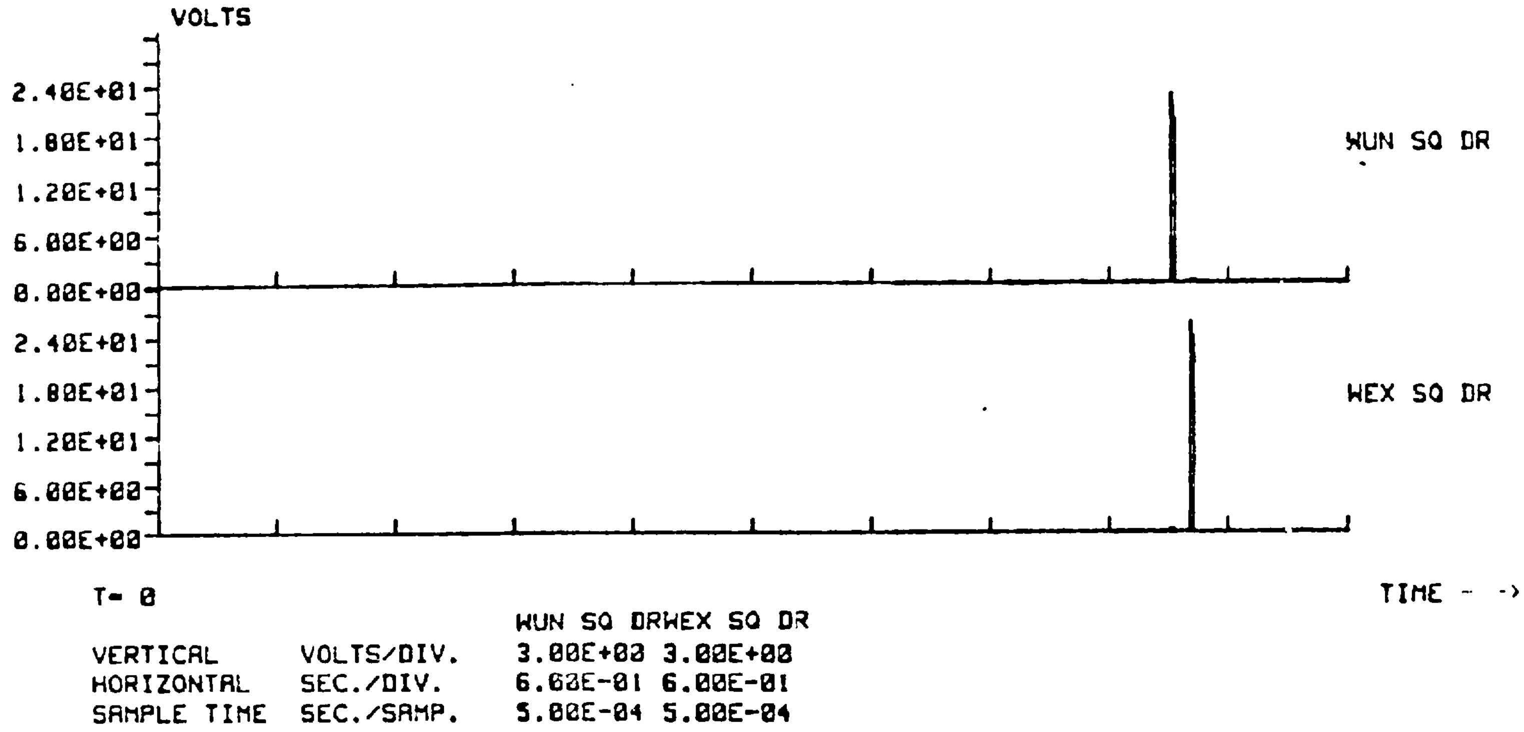


Figure 5. Window 3 Squib_driv data

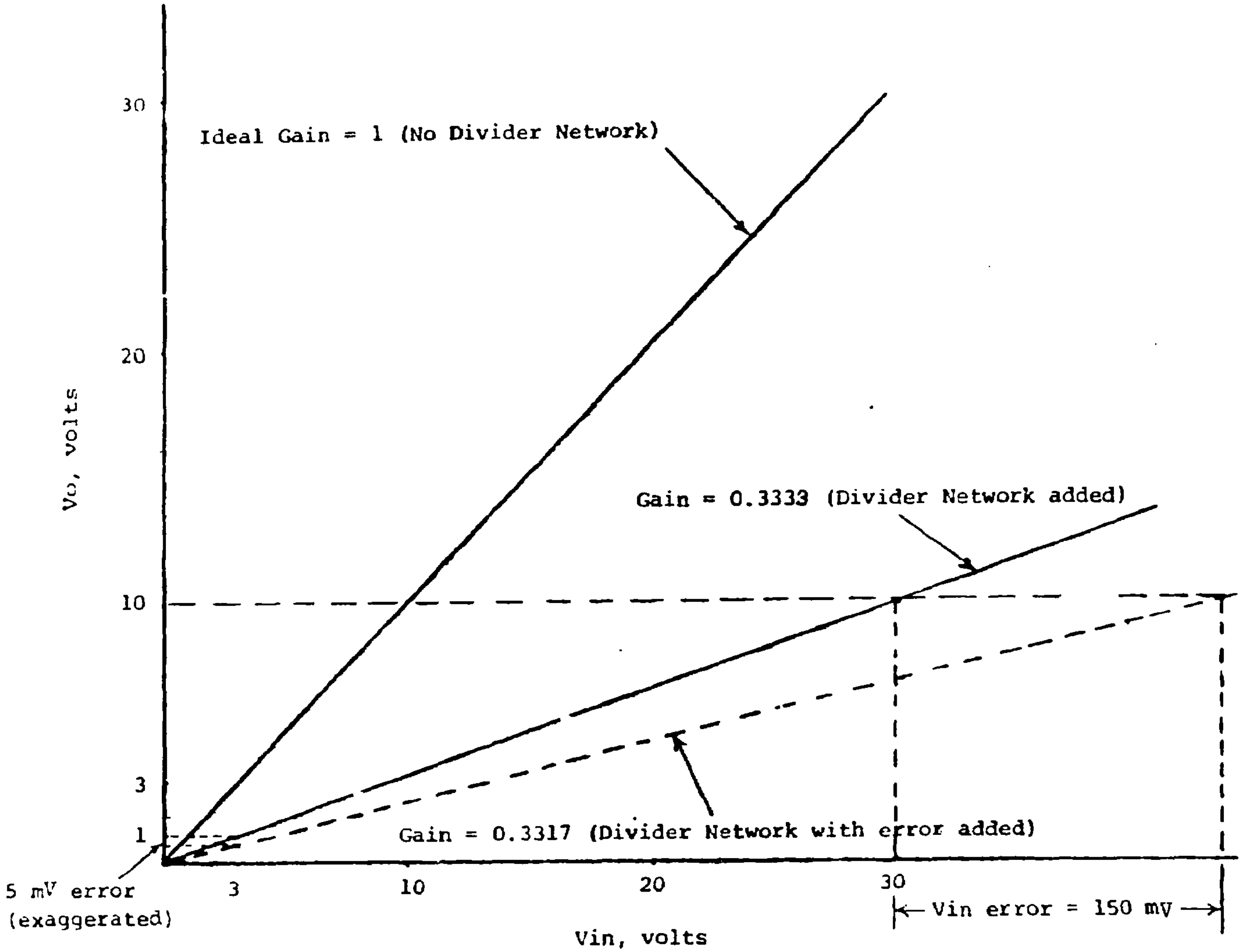


Figure 6. Low level data interface gain effects

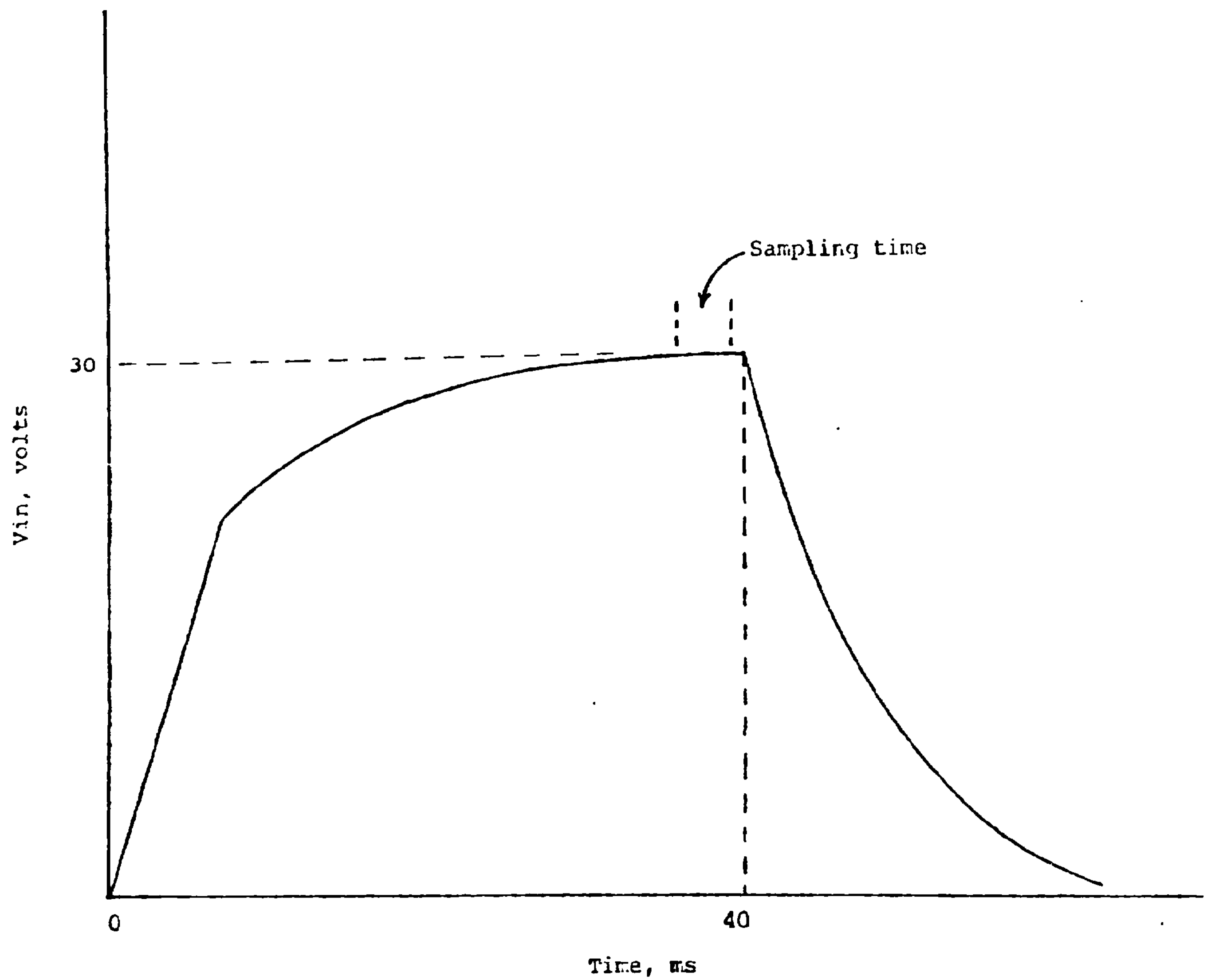


Figure 7. Divider network calibration sampling

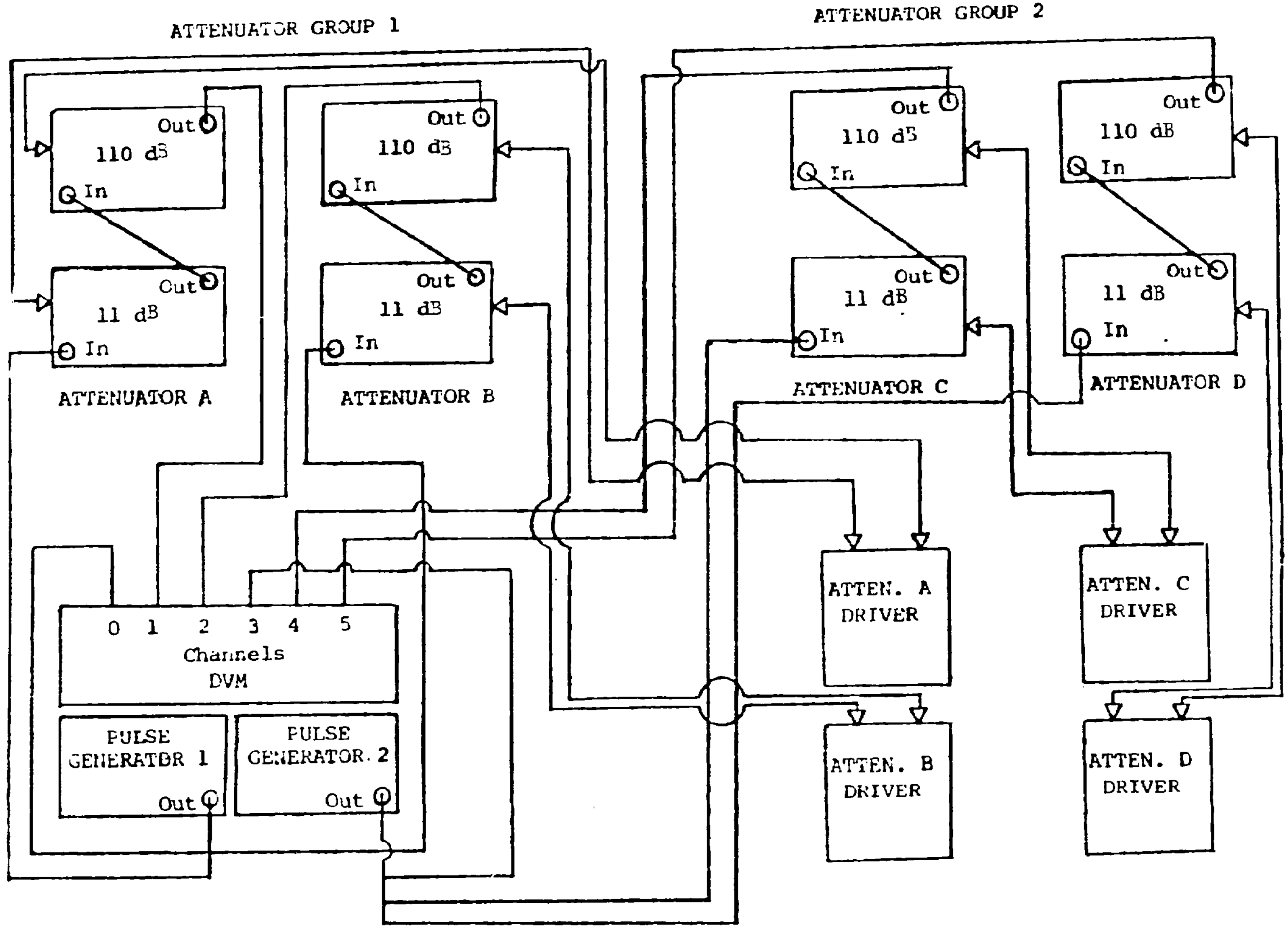


Figure 8. Gaussian calibration circuit

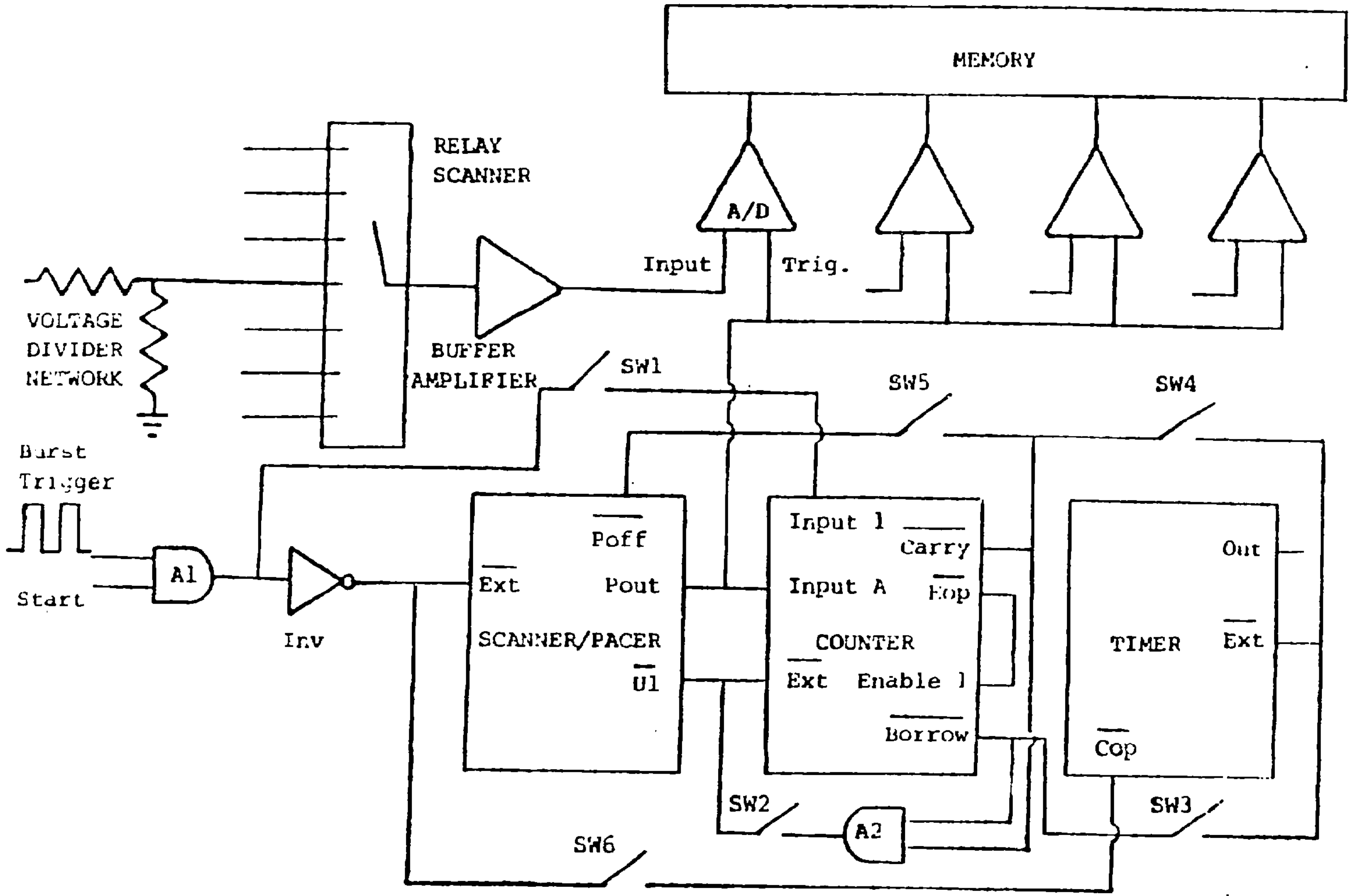


Figure 9. Burst sampling configuration

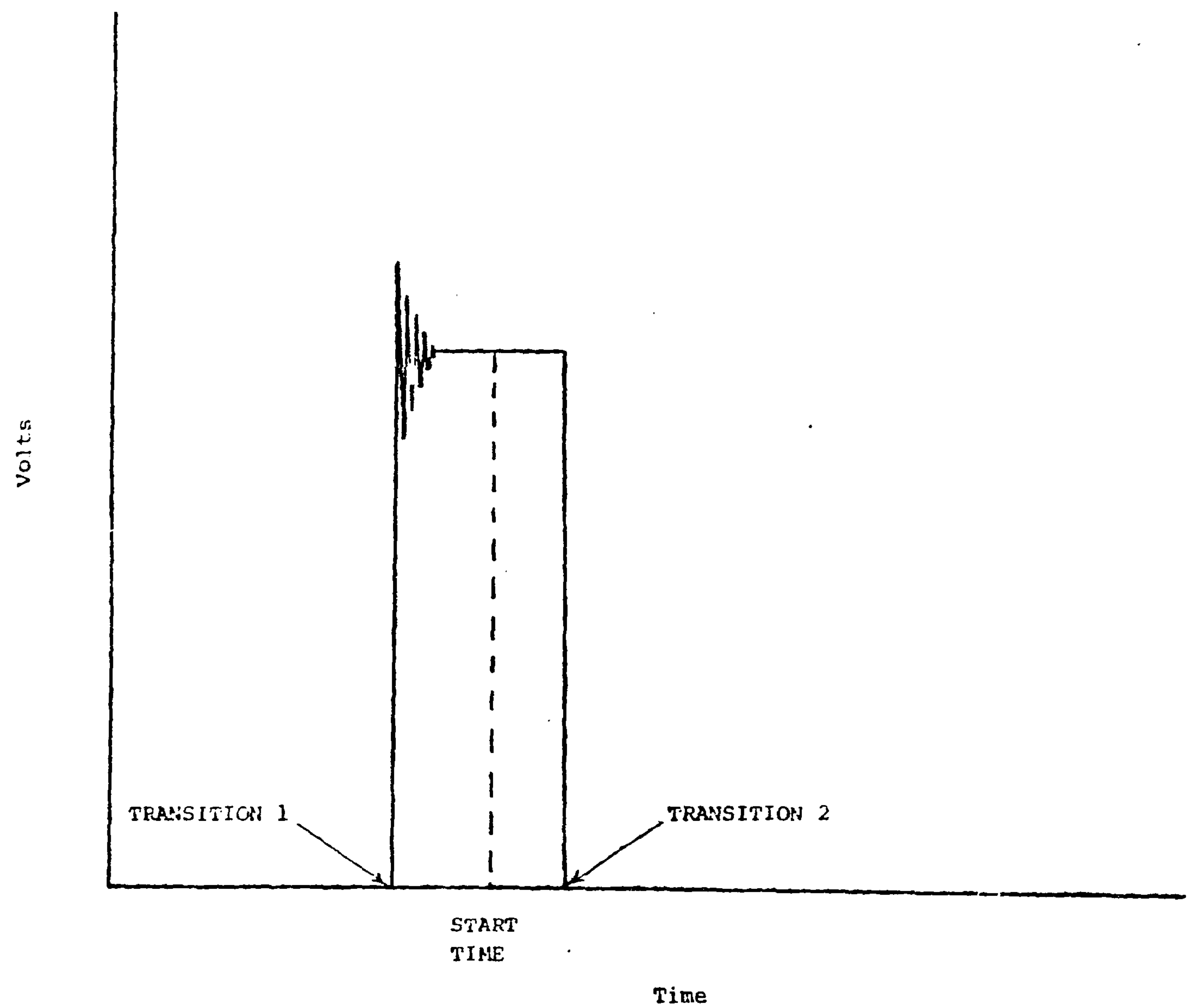


Figure 10. Typical Analyze level measurement

APPENDIX

SUBROUTINE LISTINGS

PROGRAM SUBROUTINE INDEX

```
10     SUB Test_frame
6800   SUB Utility_switch
7960   SUB Squib_driv                 ! PAR 6B
8780   SUB Ad_ld_cal
10800 SUB Drv_ps1c_rd_dvm
10848 SUB Set_db_out(Atten_grp,Dba,Dbb,Test_no)
11002 SUB Burst_setup(Trigs1,Idle_time_1)
11121    SUB Store_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
11145    SUB Get_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
11170 SUB Squib_driv(Analyze)      ! PAR 6B
```

```
10    SUB Test_frame
20    !
30    ! ==============================================================
40    !          =  14752A TEST PROCEEDURE SKELETON   =
50    !    CONSISTING OF THREE VARIABLE DATA COLLECTION WINDOWS
60    !    AND TWO IDLE WINDOWS USED FOR CARD PARAMETER ADJUSTMENTS.
70    ! ==============================================================
80    !
90    COM /Names/ Scanner,Cnt_tot,A_d1,A_d2,Multi_buffer,Timer_pacer,Timer_1,Rel
ay_d1_s7,Relay_d1_s8,Relay_d1_s9,Relay_d1_s10
100   COM /Names/ A_d3,A_d4,D_a1,D_a2,D_a3,D_a4
110   COM /Test_config/ Time_per_chan_1,Time_per_chan_2,Time_per_chan_3,Window_1
,Window_2,Window_3,Name$(20)[10],W1l1,W1l2,W2l1,W2l2,W2l3,W2l4,W3l1,W3l2
120   COM /Test_config/ Chans_per_scn_1,Chans_per_scn_2,Chans_per_scn_3,Idle_tim
e_1,Idle_time_2
130   COM /Relays/ Reela_d1_s/(*),Reela_d1_s8(*),Reela_d1_s9(*),Reela_d1_s10(*)
140   COM /Switch/ Sw_state(4),Srh,Sw_windows,Wait_gate(3)
150   COM /Test_select/ Fil$(1:72)[80],Mark$[2],Test_name$[80]
160   COM /Scanner/ Numb_of_scans,Chanls_per_scan,Sampling_rate,Time_per_chanl,C
hannels(100),Ranges(100),Intr_flag Test_period,First_scan,Last_scan,Coef(1)
170   COM /Keep/ Igrfcnt,J_chan_low,V1_lngth
180   COM /Volts_data/ Volts1(20000,1),Volts2(20000,1),Volts2b(20000,1),Volts3(2
0000,1),Dat_cnt(4)
190   COM /Cal_fact/ Cal_gain(20,20),Cal_off(20,20),J_chan_pt,J_chan_beg,Dvm_vol
ts(100),Ad_volts(110),Load_gain(130),Ircnt
200   DIM Wind_start(4)
210   Wind_start(1)=0
220   ALLOCATE Clear_lin$[75]
230   Clear_lin$=RPT$(" ",74)
240   Chanls_per_scan=1
250   Igrfcnt=0
260   J_chan_low=0
270   Tot_samples=20000
280   Numb_of_scans=Tot_samples/Chanls_per_scan
290   Sign=-1
300   Range=10
310   Set_range(A_d1,Range)
320   Set_range(A_d2,Range)
330   Set_range(A_d3,Range)
340   Set_range(A_d4,Range)
350   !
360   Volts_per_bit=Range*5.E-4
370   ALLOCATE Volts(Numb_of_scans+1000,Chanls_per_scan-1)
380   ALLOCATE Vfilt(20100,1)
390   Set_switches(Scanner,1)
400   !
410   ! GET COUNTER TOTALIZER COUNTS FOR THE THREE DATA COLLECTION WINDOWS.
420   !
430 Set_trigs: !
440   Trigs=Window_1/Time_per_chan_1
450   IF ((INT(Trigs) MOD INT(Chanls_per_scan))<>0) THEN
460   PRINT
470   PRINT "ERROR - THE WINDOW REQUESTED CANNOT GENERATE AN INTEGER NUMBER OF"
480   PRINT "        SCANS.  THE WINDOW SPECIFIED MUST BE A MULTIPLE OF THE "
490   PRINT "        THE TIME REQUIRED TO COMPLETE ONE FULL SCAN."
```

```
500   PRINT
510   STOP
520   END IF
530   Trigs1=65536-Trigs
540   !
550   Trigs=Window_2/Time_per_chan_2
560   IF ((INT(Trigs) MOD INT(Chans_per_scn_2))<>0) THEN
570   PRINT
580   PRINT "ERROR - THE WINDOW REQUESTED CANNOT GENERATE AN INTEGER NUMBER OF"
590   PRINT "        SCANS.  THE WINDOW SPECIFIED MUST BE A MULTIPLE OF THE "
600   PRINT "        THE TIME REQUIRED TO COMPLETE ONE FULL SCAN."
610   PRINT
620   STOP
630   END IF
640   Trigs2=65536-Trigs
650   !
660   Trigs3=Window_3/Time_per_chan_3
670   IF ((INT(Trigs3) MOD INT(Chanls_per_scan))<>0) THEN
680   PRINT
690   PRINT "ERROR - THE WINDOW REQUESTED CANNOT GENERATE AN INTEGER NUMBER OF"
700   PRINT "        SCANS.  THE WINDOW SPECIFIED MUST BE A MULTIPLE OF THE "
710   PRINT "        THE TIME REQUIRED TO COMPLETE ONE FULL SCAN."
720   PRINT
730   STOP
740   END IF
750   Trigs3=65536-Trigs3
760 !
770 ! SET COUNTER TOTALIZER TO FIRST WINDOW COUNT.
780 !
790   Output(Cnt_tot,Trigs1)
800 !
810 ! SET TIMER/PACER TO FIRST IDLE WINDOW.
820 !
830   Preset(Timer_pacer,Idle_time_1)
840  !
850   ! DISABLE A/Ds 3 AND 4
860   !
870   Disable_extrig(A_d3)
880   Disable_extrig(A_d4)
890   Disable_mxctrl(A_d3)
900   Disable_mxctrl(A_d4)
910   !
920   ! SET UP MEMORY AND THE SCANNER FOR THE FIRST DATA COLLECTION WINDOW.
930   !
940   Write_dif_count(Multi_buffer,0)
950   Write_wpointer(Multi_buffer,0)
960   Write_rpointer(Multi_buffer,61000)
970   Set_ext_pacing(Scanner)
980   Set_crosspoint(Scanner)
990   Set_switches(Scanner,0)
1000  Write_start_chn(Scanner,W1l1)
1010  Set_switches(Scanner,1)
1020  Write_start_chn(Scanner,W1l2)
1030  Set_switches(Scanner,1)
1040  Write_start_chn(Scanner,W2l3)
1050  Set_switches(Scanner,1)
1060  Write_start_chn(Scanner,W2l4)
1070  Set_switches(Scanner,1)
1080  Set_period(Scanner,Time_per_chan_1)
1090  !
1100  GOTO No_old_data    !   REMOVE TO USE OLD RUN_DATA REVOVERY
1110  PRINT
1120  PRINT "TYPE ENTER TO START THE SCANNER OR 'R' AND ENTER TO RE-STORE"
1130  PRINT "DATA FROM THE PREVIOUS RUN."
```

```
1140  Ans$=""
1150  INPUT Ans$
1160   IF (Ans$<>"") THEN
1170   GOTO Data_out
1180   END IF
1190 No_old_data:   !
1200  !
1210    Burst=0
1220  !
1230  ! CHECK FOR BURST MODE.
1240  !
1250    IF (Srh=11) THEN
1260    Burst=1
1270    CALL Burst_setup(Trigs1,Idle_time_1)
1280    END IF
1290  CALL Set_rela_d1_s7(10,1) ! CLOSE TIMER COPNOT TO SCANNER EXTNOT
1300  WAIT 2.0E-2
1310   SELECT Srh
1320   CASE 0,11    !,111
1330  CALL Set_rela_d1_s7(12,0) !EXTRA PROTECT TO KEEP COM2 AND SRH FROM SCANNER
1340  WAIT 2.0E-2
1350  CALL Set_rela_d1_s10(16,0) ! OPEN MASTER RESET
1360  WAIT 2.0E-2
1370  Start(Scanner)
1380   CASE ELSE
1390  CALL Set_rela_d1_s10(16,0) ! OPEN MASTER RESET
1400  WAIT 2.0E-2
1410   CALL Set_rela_d1_s7(9,1)   ! START SCANNER AND SRH
1420   CALL Set_rela_d1_s7(7,0)   ! OPEN SRH GROUNDS
1430   CALL Set_rela_d1_s7(8,0)
1440   END SELECT
1450    WAIT 1.5E-2    ! WAIT FOR RELAYS TO FINISH THEIR TRANSITIONS
1460    WAIT (Window_1)*.666*Wait_gate(1)
1470   IF (Sw_windows>12) THEN
1480   CALL Utility_switch ! CHANGES ATTENUATORS IF SWITCH HAS BEEN ENABLED IN
1490                       ! DRIVER ROUTINE.
1500   Sw_windows=Sw_windows-100
1510   END IF
1520  CALL Set_rela_d1_s7(12,0) ! KEEPS COM2 AND SRH FROM SCANNER EXT-NOT AFTER
1530                            ! THE START OF THE TEST VIA SRH
1540 Spin1:!
1550       Check_done(Cnt_tot,Cnt_flag)       ! CHECK FOR COUNT COMPLETION
1560       IF (Cnt_flag<>1) THEN
1570       GOTO Spin1
1580       ELSE
1590       END IF
1600  !
1610  ! FIRST DATA COLLECTION WINDOW IS COMPLETE.
1620  !
1630   Read_wpointer(Multi_buffer,Ans)
1640   Wind_start(2)=Ans
1650   !
1660! RESET COUNTER TOTALIZER TO SECOND WINDOW COUNT, SCANNERS TO NEW
1670! POSITIONS, AND SAMPLING RATE TO ITS NEW VALUE.
1680 !
1690  Output(Cnt_tot,Trigs2)
1700  Set_switches(Scanner,0)
1710  Write_start_chn(Scanner,W2(1)
1720  Set_switches(Scanner,1)
1730  Write_start_chn(Scanner,W2(2)
1740  Set_switches(Scanner,1)
1750  Write_start_chn(Scanner,W2(3)
1760  Set_switches(Scanner,1)
1770  Write_start_chn(Scanner,W2(4)
```

```
1780  Set_switches(Scanner,1)
1790  Set_period(Scanner,Time_per_chan_2)
1800!
1810  ! ENABLE A/Ds 3 AND 4
1820  !
1830  Enable_extrig(A_d3)
1840  Enable_extrig(A_d4)
1850  Enable_mxctrl(A_d3)
1860  Enable_mxctrl(A_d4)
1870  !
1880 Spin2:!
1890       Check_done(Timer_pacer,Tp_flag)
1900       IF (Tp_flag<>1) THEN
1910       GOTO Spin2
1920       ELSE
1930   !   PRINT "IDLE WINDOW 1 IS COMPLETE."
1940       END IF
1950       !
1960! RESET TIMER PACER TO SECOND IDLE WINDOW VALUE.
1970!
1980  Preset(Timer_pacer,Idle_time_2)
1990  Time_per_chanl=Time_per_chan_2
2000  !
2010  Wait_time=Window_2
2020 WAIT Wait_time*.66*Wait_gate(2)
2030   IF (Sw_windows>2) THEN
2040   CALL Utility_switch ! CHANGES ATTENUATORS IF SWITCH HAS BEEN ENABLED IN
2050                       ! DRIVER ROUTINE.
2060   Sw_windows=Sw_windows-10
2070   END IF
2080 Spin3:!
2090       Check_done(Cnt_tot,Cnt_flag)       ! CHECK FOR COUNT COMPLETION
2100       IF (Cnt_flag<>1) THEN
2110       GOTO Spin3
2120       ELSE
2130       END IF
2140       !
2150  !
2160  ! SECOND DATA COLLECTION WINDOW IS COMPLETE.
2170  ! SECOND IDLE WINDOW IS ASSUMED TO BE IN EFFECT.
2180  ! NOW SET UP THE SCANNER CARD FOR THE THIRD DATA COLLECTION WINDOW.
2190  !
2200  ! DISABLE A/Ds 3 AND 4
2210  !
2220  Disable_extrig(A_d3)
2230  Disable_extrig(A_d4)
2240  Disable_mxctrl(A_d3)
2250  Disable_mxctrl(A_d4)
2260  !
2270  Output(Cnt_tot,Trigs3)
2280  Set_switches(Scanner,0)
2290  Write_start_chn(Scanner,W3l1)
2300  Set_switches(Scanner,1)
2310  Write_start_chn(Scanner,W3l2)
2320  Set_switches(Scanner,1)
2330  Set_period(Scanner,Time_per_chan_3)
2340  !
2350   Read_wpointer(Multi_buffer,Ans)
2360   Wind_start(3)=Ans
2370 Spin4:!
2380       Check_done(Timer_pacer,Tp_flag)
2390       IF (Tp_flag<>1) THEN
2400       GOTO Spin4
2410       ELSE
```

```
2420      PRINT "IDLE WINDOW 2 IS COMPLETE."
2430      END IF
2440      !
2450  Wait_time=Window_3
2460  !
2470  WAIT Wait_time*.66*Wajt_gate(3) ! ALLOW TIME FOR THE THIRD DATA WINDOW TO
COLLECT DATA.
2480  IF (Sw_windows>0) THEN
2490   CALL Utility_switch ! CHANGES ATTENUATORS IF SWITCH HAS BEEN ENABLED IN
2500                       ! DRIVER ROUTINE.
2510   Sw_windows=0
2520  END IF
2530  !
2540!
2550 Spin5:!
2560      Check_done(Cnt_tot,Cnt_flag)      ! CHECK FOR COUNT COMPLETION
2570      IF (Cnt_flag<>1) THEN
2580      GOTO Spin5
2590      ELSE
2600      END IF
2610      !
2620      Enable_lockout(Multi_buffer)
2630  !
2640  !  THE TEST IS OVER.
2650  !
2660  Wait_time=Window_1+Idle_time_1+Window_2+Idle_time_2+Window_3
2670  Start_x=0
2680  Stop_x=Wait_time
2690  Start_x_old=0
2700  Stop_x_old=Wait_time
2710  !
2720  PRINT "SCANNER HAS BEEN STOPPED."
2730  !
2740  CALL Set_rela_d1_s7(10,0)   !  T/P COPNOT TO SCAN EXTNOT
2750  WAIT 2.0E-2
2760  Stop(Scanner)
2770  WAIT 1.00E-1
2780  Check_done(Timer_pacer,Dum)
2790  Check_done(Cnt_tot,Dum)
2800  Stop(Scanner)
2810  CALL Disab_sups_da
2820  Set_switches(Scanner,0)
2830  !
2840  ! PREPARE FOR AN UPLOADING OF DATA FROM THE 6944.
2850  !
2860  Read_rpointer(Multi_buffer,Rcnt)
2870  PRINT "READ POINT = ",Rcnt
2880  Read_wpointer(Multi_buffer,Wcnt)
2890  PRINT "WRITE POINT = ",Wcnt
2900  Read_dif_count(Multi_buffer,Cnt)
2910  PRINT "DIF COUNT = ",Cnt
2920! ALLOCATE Volts(999,2)
2930  Set_fifo_in(Multi_buffer)
2940  Write_rpointer(Multi_buffer,0)
2950  Write_wpointer(Multi_buffer,64000)
2960  CONTROL 32,1;0
2970  Mux_no=2
2980  Tot_samples=((Window_1/Time_per_chan_1)*2+(Window_2/Time_per_chan_2)*4+(Wi
ndow_3/Time_per_chan_3)*2)
2990  PRINT "TOTAL SAMPLES NEEDED = ",Tot_samples
3000  Tot_samples=20000
3010  Dat_cnt(1)=(2*Window_1/(Time_per_chan_1*Chans_per_scn_1))-1
3020  Dat_cnt(2)=(4*Window_2/(Time_per_chan_2*Chans_per_scn_2))-1
3030  Dat_cnt(3)=(2*Window_3/(Time_per_chan_3*Chans_per_scn_3))-1
```

```
3650  IF (Ans$="B") THEN Temp=2
3660  IF (Ans$="C") THEN Temp=4       !TEMP=6   FOR AD-2
3670         IF (Ans$="D") THEN Temp=6
3680  GOTO Hop_over
3690   END IF
3700  PRINT TABXY(1,18);Clear_lin$
3710 Rep: !
3720  PRINT TABXY(1,19);Clear_lin$
3730  PRINT TABXY(1,19);"TYPE IN THE NUMBER OF THE CHANNEL YOU WISH TO PLOT."
3740  INPUT Ans$
3750  Test=NUM(Ans$)
3760  IF (Test<48 OR Test>57) THEN
3770  PRINT TABXY(1,18);"A NUMBER MUST BE SUPPLIED AT THIS TIME."
3780  GOTO Rep
3790  END IF
3800  Temp=VAL(Ans$)
3810 Hop_over:!
3820  J_chan_strt=Temp
3830  J_chan_stop=Temp
3840  Chanls_per_scan=Chans_per_scn_2
3850  Time_per_chanl=Time_per_chan_2
3860  J_chan_pt=2
3870  T_off_set=Window_1+Idle_time_1
3880  IF (J_chan_strt<2) THEN
3890  Chanls_per_scan=Chans_per_scn_1
3900  Time_per_chanl=Time_per_chan_1
3910  J_chan_pt=0
3920  T_off_set=0
3930  END IF
3940        IF (J_chan_strt>3) THEN
3950        J_chan_pt=4
3960        T_off_set=Window_1+Idle_time_1
3970        END IF
3980  IF (J_chan_strt>5) THEN
3990  J_chan_pt=6
4000  T_off_set=Window_1+Idle_time_1+Window_2+Idle_time_2
4010  Time_per_chanl=Time_per_chan_3
4020  Chanls_per_scan=Chans_per_scn_3
4030  END IF
4040    SELECT J_chan_pt
4050    CASE =0
4060  Rec_num=1
4070    CASE 2,4
4080  Rec_num=6
4090  IF (Dat_cnt(1)<=20) THEN Rec_num=1
4100    CASE =6
4110  Rec_num=11
4120  IF (Dat_cnt(1)<=20) THEN Rec_num=Rec_num-5
4130  IF (Dat_cnt(2)<=20) THEN Rec_num=Rec_num-5
4140    END SELECT
4150  IF (Srh=1) THEN ! CHECK FOR SCANNED AND MUXED CASE WITH ONE 60K
4160                    ! COLLECTION WINDOW.
4170    SELECT J_chan_pt
4180    CASE 0,1,6,7
4190    T_off_set=T_off_set+Time_per_chanl*1000
4200    CASE ELSE
4210    T_off_set=T_off_set+Time_per_chanl*500
4220    END SELECT
4230    Ta_start=0
4240    Ta_stop=T_off_set-Time_per_chanl*Chanls_per_scan
4250    Tb_start=T_off_set
4260    Tb_stop=2*T_off_set-Time_per_chanl*Chanls_per_scan
4270    Tc_start=2*T_off_set
4280    Tc_stop=3*T_off_set-Time_per_chanl*Chanls_per_scan
```

```
3040 !
3050 !  CREATE RUN_DATA AT THE CORRECT LENGTH
3060 !
3070  PURGE "/FTM/HOME_DIR00/RUN_DATA:,1400"
3080  Recs=0
3090  Itop=3
3100  IF (Srh=1) THEN Itop=3
3110  FOR Krec=1 TO Itop
3120  IF (Dat_cnt(Krec)>20) THEN Recs=Recs+1 ! OR SRH=1 ALSO ?
3130  NEXT Krec
3140  Recs=Recs*5
3150  CREATE BDAT "/FTM/HOME_DIR00/RUN_DATA:,1400",Recs,32100
3160 !
3170 IF (Burst=1) THEN
3180 Wind_start(1)=0
3190 Wind_start(2)=20000
3200 Wind_start(3)=40000
3210 END IF
3220! IF (Srh=12 OR Srh=13) THEN
3230! Wind_start(1)=0
3240! Wind_start(2)=Dat_cnt(1)
3250! Wind_start(3)=Dat_cnt(1)+Dat_cnt(2)
3260! END IF
3270 Kpas=0
3280 Itop=3
3290 IF (Srh=1) THEN Itop=3
3300 FOR Ipas=1 TO Itop
3310   IF ((Dat_cnt(Ipas)>20) OR (Srh=1)) THEN  ! SKIP OVER IF DATA WINDOW IS A
 DUMMY
3320 Kpas=Kpas+1
3330 Rec_num=(Kpas-1)*5+1
3340 Write_rpointer(Multi_buffer,Wind_start(Ipas))
3350 IF (Srh=1) THEN
3360  Write_rpointer(Multi_buffer,(Ipas-1)*20000)
3370 END IF
3380 !
3390 !  UPLOAD DATA
3400 !
3410 Input_rblock(Multi_buffer,Volts(*),Tot_samples) ! STORED WITH RIGHT VARIES
 MOST
3420 CALL Store_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
3430   END IF
3440 NEXT Ipas
3450 CONTROL 32,1;1
3460 Mcnt=0
3470 Data_out: !
3480 Dat_cnt(1)=(2*Window_1/(Time_per_chan_1*Chans_per_scn_1))-1
3490 Dat_cnt(2)=(4*Window_2/(Time_per_chan_2*Chans_per_scn_2))-1
3500 Dat_cnt(3)=(2*Window_3/(Time_per_chan_3*Chans_per_scn_3))-1
3510 PRINT TABXY(1,18);Clear_lin$
3520 PRINT TABXY(1,19);Clear_lin$
3530 PRINT TABXY(1,20);Clear_lin$
3540 !
3550 PRINT TABXY(1,18);"DO YOU WISH TO (L)LIST OR (P)PLOT THE DATA ?";
3560 !
3570 INPUT Path$
3580  IF (Path$="L") THEN
3590 PRINT TABXY(1,18);Clear_lin$
3600 PRINT TABXY(1,18);"TYPE IN THE LETTER OF THE CHANNEL GROUP YOU WISH TO LIS
T."
3610        PRINT "(A)=0,1  (B)=2,3   (C)=4,5    (D)=6,7" !INSTEAD OF LINE
3620                                                     !ABOVE   AD-2
3630 INPUT Ans$
3640 Temp=0
```

```
4290    PRINT
4300    PRINT "SCANNED/MUXED DATA IS VIEWED IN THE FOLLOWING GROUPS:"
4310    PRINT
4320    PRINT "    GROUP              START TIME          STOP TIME"
4330    PRINT "    -----              ----------          ---------"
4340    PRINT "      A            ",Ta_start,"        ",Ta_stop
4350    PRINT "      B            ",Tb_start,"        ",Tb_stop
4360    PRINT "      C            ",Tc_start,"        ",Tc_stop
4370    PRINT
4380    PRINT "TYPE THE LETTER OF THE GROUP DESIRED FOLLOWED BY 'ENTER'."
4390    INPUT Anss$
4400    Mux_scan_cnt=0
4410    IF (Anss$="B") THEN Mux_scan_cnt=1
4420    IF (Anss$="C") THEN Mux_scan_cnt=2
4430    Rec_num=Rec_num+5*Mux_scan_cnt
4440  END IF
4450    IF (Rec_num_old<>Rec_num) THEN
4460    Rec_num_old=Rec_num
4470  CALL Get_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
4480 !
4490 ! GET CALIBRATION DATA IF THIS IS NOT A CALIBRATION TEST
4500 !
4510  IF (Sw_state(2)<>1000000) THEN
4520   ASSIGN @File TO "/FTM/HOME_DIR00/CAL_FACT:,1400"
4530   ENTER @File;Cal_gain(*),Cal_off(*),Load_gain(*)
4540  ELSE
4550   FOR Ia=1 TO 4
4560   FOR Ib=1 TO 4
4570   Cal_gain(Ia,Ib)=1
4580   Cal_off(Ia,Ib)=0
4590   NEXT Ib
4600   NEXT Ia
4610   FOR Ia=1 TO 130
4620   Load_gain(Ia)=1
4630   NEXT Ia
4640  END IF
4650 !
4660  Wait_time=Window_1+Idle_time_1+Window_2+Idle_time_2+Window_3
4670  !   Data_in:  !
4680  !
4690  ! MANIPULATE UPLOADED DATA
4700  !
4710  FOR J=0 TO Chanls_per_scan-1
4720  FOR I=0 TO Numb_of_scans-1
4730    Volts(I,J)=Sign*Volts(I,J)*Volts_per_bit
4740  NEXT I
4750  NEXT J
4760 !
4770 !  BREAK UP VOLTS DATA INTO VOLTS1, VOLTS2, AND VOLTS3 ARRAYS.
4780 !
4790  V1_lngth=20000       !6500
4800 !
4810 !  SET VOLTS1 TO 123.123
4820 !
4830 !
4840 !  SET THE VOLTS2 AND VOLTS3 ARRAYS TO 123.123
4850 !
4860  Ncnt=(2*Window_1/(Time_per_chan_1*Chans_per_scn_1))-1
4870  Ncntnew=(4*Window_2/(Time_per_chan_2*Chans_per_scn_2))-1
4880  Nfinal=(2*Window_3/(Time_per_chan_3*Chans_per_scn_3))-1
4890    SELECT J_chan_pt
4900    CASE =0
4910  !
4920  !  SET ARRAY TO 123.123
```

```
4930  !
4940  FOR J=0 TO 10000          !20000
4950  Volts1(J,0)=123.123
4960  Volts1(J,1)=123.123
4970  NEXT J
4980  !
4990  Kk=-1
5000  FOR J=0 TO Ncnt STEP 2
5010  Kk=Kk+1
5020  Volts1(Kk,0)=(Volts(J,0)+Cal_gain(2,1)*Cal_off(2,1))/(Cal_gain(2,1)*Load_g
ain(W1l1))
5030  Volts1(Kk,1)=(Volts(J+1,0)+Cal_gain(2,2)*Cal_off(2,2))/(Cal_gain(2,2)*Load
_gain(W1l2))
5040  NEXT J
5050   CASE 2,4
5060 !
5070 !  SET THE VOLTS2 ARRAY TO 123.123
5080 !
5090  FOR J=0 TO 10000          !20000
5100  Volts2(J,0)=123.123
5110  Volts2(J,1)=123.123
5120          Volts2b(J,0)=123.123
5130          Volts2b(J,1)=123.123
5140  NEXT J
5150  !
5160  Kk=-1             !******************************
5170  FOR K=0 TO Ncntnew STEP 4       !STEP 4
5180 !J=Ncnt+1+K
5190  J=K
5200  Kk=Kk+1
5210  Volts2(Kk,0)=(Volts(J,0)+Cal_gain(2,1)*Cal_off(2,1))/(Cal_gain(2,1)*Load_g
ain(W2l1))
5220  Volts2(Kk,1)=(Volts(J+1,0)+Cal_gain(2,2)*Cal_off(2,2))/(Cal_gain(2,2)*Load
_gain(W2l2))
5230          Volts2b(Kk,0)=(Volts(J+2,0)+Cal_gain(4,1)*Cal_off(4,1))/(Cal_gain(4
,1)*Load_gain(W2l3))
5240          Volts2b(Kk,1)=(Volts(J+3,0)+Cal_gain(4,2)*Cal_off(4,2))/(Cal_gain(4
,2)*Load_gain(W2l4))
5250  NEXT K
5260   CASE =6
5270 !
5280 !  SET THE VOLTS3 ARRAY TO 123.123
5290 !
5300  FOR J=0 TO 10000          !20000
5310  Volts3(J,0)=123.123
5320  Volts3(J,1)=123.123
5330  NEXT J
5340  !
5350  Kk=-1             !******************************
5360  FOR K=0 TO Nfinal STEP 2
5370   J=K
5380  Kk=Kk+1
5390  Volts3(Kk,0)=(Volts(J,0)+Cal_gain(2,1)*Cal_off(2,1))/(Cal_gain(2,1)*Load_g
ain(W3l1))
5400  Volts3(Kk,1)=(Volts(J+1,0)+Cal_gain(2,2)*Cal_off(2,2))/(Cal_gain(2,2)*Load
_gain(W3l2))
5410  NEXT K
5420    END SELECT
5430    END IF     !  Rec_num<>Rec_num_old TEST
5440  Spotx=6
5450  PRINT TABXY(1,16);Clear_lin$
5460  PRINT TABXY(1,17);Clear_lin$
5470  PRINT TABXY(1,18);Clear_lin$
5480  PRINT TABXY(1,19);Clear_lin$
```

```
5490  PRINT TABXY(1,16);"TYPE IN THE TIME AT WHICH YOU WISH TO BEGIN OBSERVING T
HE DATA.  "
5500  Mcnt=Mcnt+1
5510  IF (Mcnt<>1) THEN PRINT "THE LETTER *M* WILL INVOKE THE MARKER."
5520  INPUT Ans$
5530  IF (Ans$="M") THEN
5540  CALL Mark(Spotx,Full_period,T,Start_x_old,Stop_x_old,Ans$)
5550  END IF
5560  PRINT TABXY(66,16);Ans$
5570  First_scan=INT((VAL(Ans$)-T_off_set)/(Time_per_chanl*Chanls_per_scan))
5580  Start_x=VAL(Ans$)     ! I THINK THIS CAN REPLACE THE PREVIOUS LINE
5590 Ask_again:!
5600  PRINT TABXY(1,17);Clear_lin$
5610  PRINT TABXY(1,18);Clear_lin$
5620  PRINT TABXY(1,19);Clear_lin$
5630  PRINT TABXY(1,17);"TYPE IN THE TIME AT WHICH YOU WISH TO STOP OBSERVING TH
E DATA.   "
5640  IF (Mcnt<>1) THEN PRINT "THE LETTER *M* WILL INVOKE THE MARKER."
5650  INPUT Ans$
5660  IF (Ans$="M") THEN
5670  CALL Mark(Spotx,Full_period,T,Start_x_old,Stop_x_old,Ans$)
5680  END IF
5690  Stop_x=VAL(Ans$)
5700  IF (Stop_x>Wait_time) THEN
5710  PRINT TABXY(1,18);Clear_lin$
5720  PRINT TABXY(1,18);"TIME REQUESTED EXCEEDS DATA LIMIT !!  TYPE ENTER TO TRY
 AGAIN.";
5730  INPUT Ans$
5740  GOTO Ask_again
5750  END IF
5760  PRINT TABXY(66,17);Ans$
5770  Stop_x_old=Stop_x
5780  Start_x_old=Start_x
5790  Last_scan=INT((VAL(Ans$)-T_off_set)/(Time_per_chanl*Chanls_per_scan))
5800  IF (Last_scan>Numb_of_scans-1) THEN Last_scan=Numb_of_scans-1
5810  IF (Path$<>"L") THEN GOTO Plt
5820  PRINT TABXY(1,18);Clear_lin$
5830  PRINT TABXY(1,18);"DO YOU WISH THE LIST TO GO TO THE (P)PRINTER OR (C)CRT
?"
5840  INPUT Ans$
5850  IF (Ans$="P") THEN PRINTER IS 9
5860  OUTPUT KBD;CHR$(255)&"K";
5870  T=Time_per_chanl*(Chanls_per_scan*First_scan-1)+T_off_set
5880  PRINT
5890  PRINT "                   "&Name$(J_chan_pt)&"                  "&Name$(J_chan
_pt+1)&"                 "&Name$(J_chan_pt+2)
5900  PRINT
5910  PRINT "                 TIME       VALUE          TIME      VALUE           TIME
VALUE"
5920  PRINT
5930  FOR Ii=First_scan TO Last_scan
5940  T=T+Time_per_chanl
5950  IF (Ii<0) THEN GOTO Nxt_ii
5960  T1=T+(Chanls_per_scan-1)*Time_per_chanl
5970  T2=T1+(Chanls_per_scan-1)*Time_per_chanl
5980  Scan$="SCAN"
5990  IF (J_chan_pt=0) THEN
6000  PRINT USING "4A,X,5D,2X,S1D.3DE,X,S1D.3DE,2X,S1D.3DE,1X,S1D.3DE,2X,S1D.3DE
,1X,S1D.3DE";Scan$,Ii,T,Volts1(Ii,0),T1,Volts1(Ii,1)
6010  T=T1
6020  END IF
6030  IF (J_chan_pt=2) THEN
6040  PRINT USING "4A,X,5D,2X,S1D.3DE,X,S1D.3DE,2X,S1D.3DE,1X,S1D.3DE,2X,S1D.3DE
,1X,S1D.3DE";Scan$,Ii,T,Volts2(Ii,0),T1,Volts2(Ii,1)
```

```
6050  T=T2
6060  END IF
6070      IF (J_chan_pt=4) THEN
6080      PRINT USING "4A,X,5D,2X,S1D.3DE,X,S1D.3DE,2X,S1D.3DE,1X,S1D.3DE,2X,S1D
.3DE,1X,S1D.3DE";Scan$,Ii,T,Volts2b(Ii,0),T1,Volts2b(Ii,1)
6090      T=T2
6100      END IF
6110  IF (J_chan_pt=6) THEN
6120  PRINT USING "4A,X,5D,2X,S1D.3DE,X,S1D.3DE,2X,S1D.3DE,1X,S1D.3DE,2X,S1D.3DE
,1X,S1D.3DE";Scan$,Ii,T,Volts3(Ii,0),T1,Volts3(Ii,1)
6130  T=T2
6140  END IF
6150 Nxt_ii:  !
6160  NEXT Ii
6170  PRINTER IS 1
6180  GOTO Quest
6190 Plt:  !
6200  Vert_min=0.
6210  PRINT TABXY(1,18);Clear_lin$
6220  PRINT TABXY(1,18);"TYPE IN THE MINIMUM VALUE OF THE VERTICAL AXIS. (DEFAUL
T = 0.0)  ";
6230  Ans$=""
6240  INPUT Ans$
6250  IF (Ans$<>"") THEN
6260  Vert_min=VAL(Ans$)
6270  END IF
6280  PRINT VAL$(Vert_min)
6290  Vert_max=Range
6300  Def_val$=VAL$(Range)
6310  PRINT TABXY(1,19);Clear_lin$
6320  PRINT TABXY(1,19);"TYPE IN THE MAXIMUM VALUE OF THE VERTICAL AXIS. (DEFAUL
T="&Def_val$&")       ";
6330  Ans$=""
6340  INPUT Ans$
6350  IF (Ans$<>"") THEN
6360  Vert_max=VAL(Ans$)
6370  END IF
6380  PRINT VAL$(Vert_max)
6390  J_name_pt=J_chan_strt
6400  IF (J_chan_pt=0) THEN
6410  CALL Plot_channels(Volts1(*),Start_x,Vert_min,Stop_x,Vert_max,"VOLTS",0,J_
chan_strt,J_chan_stop,Name$(*),J_name_pt,Titles$,T_off_set,Scan_size)
6420  END IF
6430  IF (J_chan_pt=2) THEN
6440  J_chan_strt=J_chan_strt-2
6450  J_chan_stop=J_chan_strt
6460  CALL Plot_channels(Volts2(*),Start_x,Vert_min,Stop_x,Vert_max,"VOLTS",0,J_
chan_strt,J_chan_stop,Name$(*),J_name_pt,Titles$,T_off_set,Scan_size)
6470  END IF
6480        IF (J_chan_pt=4) THEN
6490        J_chan_strt=J_chan_strt-4
6500        J_chan_stop=J_chan_strt
6510        CALL Plot_channels(Volts2b(*),Start_x,Vert_min,Stop_x,Vert_max,"VOLTS
",0,J_chan_strt,J_chan_stop,Name$(*),J_name_pt,Titles$,T_off_set,Scan_size)
6520        END IF
6530  IF (J_chan_pt=6) THEN
6540  J_chan_strt=J_chan_strt-6
6550  J_chan_stop=J_chan_strt
6560  CALL Plot_channels(Volts3(*),Start_x,Vert_min,Stop_x,Vert_max,"VOLTS",0,J_
chan_strt,J_chan_stop,Name$(*),J_name_pt,Titles$,T_off_set,Scan_size)
6570  END IF
6580  PRINT TABXY(1,18);Clear_lin$
6590  PRINT TABXY(1,19);Clear_lin$
6600 Quest:  !
```

```
6610  PRINT TABXY(1,16);Clear_lin$
6620  PRINT TABXY(1,17);Clear_lin$
6630  PRINT TABXY(1,18);Clear_lin$
6640  PRINT TABXY(1,19);Clear_lin$
6650  PRINT TABXY(1,18);"DO YOU WISH TO LIST OR PLOT THE DATA AGAIN ?  (Y)YES OR
 (N)NO"
6660  Ans$=""
6670  INPUT Ans$
6680  IF (Ans$="N") THEN
6690  OUTPUT 2;Clear_crt$;
6700  GOTO Main_end
6710  END IF
6720  GOTO Data_out
6730  !
6740 Main_end: !
6750  ! --------- END OF MAIN PROGRAM -----------
6760  PRINT "MAIN_END"
6770  !
6780  SUBEND
6781  !
6790  !   *******************************************************************
6791  !
6800  SUB Utility_switch
6810  !
6820  ! THIS ROUTINE PROVIDES AN ALL PURPOSE SWITCH TO BE USED ONCE DURING
6830  ! EACH TEST WINDOW.  THE FOLLOWING DEFINES EACH SWITCH AND HOW IT IS
6840  ! INVOKED.
6850  !
6860  ! SW_STATE          DEFINITION
6870  !----------        --------------------
6880  !     0             ALL SWITCHES STAY OPEN.
6890  !
6900  !  1 - 30           SETS ATTENUATORS AND PULSE GEN VOLTS ACCORDING TO
6910  !                   THE VALUE OF SW_STATE.
6920  !
6930  ! 31 - 499          PRODUCES HP-IB AND OTHER VARIOUS COMMANDS.
6940  !
6950  ! 500-1000          SETS PULSE GEN PULSE WIDTHS ACCORDING TO SW_STATE.
6960  !
6970  ! 1000 -            SETS RELAY OUTPUT CARD RELAYS ACCORDING TO SW_STATE.
6980  !
6990  ! 1000000 -         DRIVES CALIBRATION INSTRUMENTATION
7000  !
7010  COM /Names/ Scanner,Cnt_tot,A_d1,A_d2,Multi_buffer,Timer_pacer,Timer_1,Rel
ay_d1_s7,Relay_d1_s8,Relay_d1_s9,Relay_d1_s10
7020  COM /Names/ A_d3,A_d4,D_a1,D_a2,D_a3,D_a4
7030  COM /Switch/ Sw_state(4),Srh,Sw_windows,Wait_gate(3)
7040  !
7050  ! DETERMINE SW_STATE POINTER.
7060  !
7070  ! Ipoint=DROUND(Sw_windows/2,1)
7080  ! Ipoint=(Ipoint MOD 4)+1
7090  !
7100    Power=LGT(Sw_windows)
7110    Ipoint=ABS(Power-3)
7120  !
7130  IF (Sw_state(Ipoint)<>0) THEN
7140      IF (Sw_state(Ipoint)<=30) THEN
7150      CALL Set_pgen_atten(Sw_state(Ipoint)) ! SETS ATTENUATORS AND PULSE GEN
 VOLTAGE.
7160      ELSE
7170  !      IF (Sw_state(Ipoint)<=30) THEN
7180  !      ! SET PULSE GEN PULSE WIDTHS
7190  !      ASSIGN @Pulse_gens TO 708,718
```

```
7200  !     Wide=(Sw_state(Ipoint)-20)*300+300        !-30
7210  !     Wide$=VAL$(Wide)
7220  !     Tre=.7*Wide
7230  !     Lee=.7*Wide
7240  !     Tre$=VAL$(Tre)
7250  !     Lee$=VAL$(Lee)
7260  !     OUTPUT @Pulse_gens;"WID"&Wide$&"NS,LEE"&Lee$&"NS,TRE"&Tre$&"NS"
7270     !  Ftic_setup("BOTH_GENS","WIDTH",Wide$&"NS")
7280  !     ELSE
7290          IF (Sw_state(Ipoint)<500) THEN      ! 1000
7300          SELECT Sw_state(Ipoint)
7310           CASE 31
7320            CALL Rotat_flop_down(20)
7330           CASE 32
7340            CALL Rotat_flop_down(-40)
7350     !     IF (Sw_state(Ipoint)=31) THEN CALL Rotat_flop_down(20)
7360     !     IF (Sw_state(Ipoint)=32) THEN CALL Rotat_flop_down(-40)
7370     !     IF (Sw_state(Ipoint)=40) THEN
7380           CASE 33 TO 38
7390           CALL Thumper(Sw_state(Ipoint))
7400           CASE 40
7410            Start(D_a4)
7420     !     END IF
7430     !     IF (Sw_state(Ipoint)=50) THEN    !  CLOSE MANUAL CAGE IN 6W
7440           CASE 50
7450            CALL Set_rela_d1_s7(15,1)
7460            CALL Set_rela_d1_s7(16,1)
7470     !     END IF
7480     !     IF (Sw_state(Ipoint)=51) THEN    !  SET TA AND YAW LOS TO ZERO
7490           CASE 51
7500            CALL Set_rela_d1_s8(4,1)          !  IN ATT_HLD
7510            CALL Set_rela_d1_s8(2,1)
7520            Ftic_setup("SET_PS1B_VOLTAG","0")
7530     !     END IF
7540     !     IF (Sw_state(Ipoint)=52) THEN     ! USED BY GUID GAIN
7550           CASE 52
7560            Ftic_setup("SET_PS1B_VOLTAG","0")
7570            CALL Set_rela_d1_s8(10,1)
7580     !     END IF
7590          END SELECT
7600          ELSE
7610            IF (Sw_state(Ipoint)<1000) THEN
7620       ! SET PULSE GEN PULSE WIDTHS
7630       ASSIGN @Pulse_gens TO 708,718
7640       Wide=(Sw_state(Ipoint)-500)*300+300
7650       Wide$=VAL$(Wide)
7660       Tre=.7*Wide
7670       Lee=.7*Wide
7680       Tre$=VAL$(Tre)
7690       Lee$=VAL$(Lee)
7700       OUTPUT @Pulse_gens;"WID"&Wide$&"NS,LEE"&Lee$&"NS,TRE"&Tre$&"NS"
7710     ! Ftic_setup("BOTH_GENS","WIDTH",Wide$&"NS")
7720            ELSE
7730            IF (Sw_state(Ipoint)<1000000) THEN
7740   !
7750   ! SW_STATE MUST BE A 4 OR 5 DIGIT NUMBER  12       34        5
7760   !                                       SLOT    RELAY    STATE
7770   !
7780            Slot=INT(Sw_state(Ipoint)/1000)
7790            Rel_num=INT((Sw_state(Ipoint)-Slot*1000)/10)
7800            State=Sw_state(Ipoint)-(Slot*1000+Rel_num*10)
7810            IF (Slot=7) THEN CALL Set_rela_d1_s7(Rel_num,State)
7820            IF (Slot=8) THEN CALL Set_rela_d1_s8(Rel_num,State)
7830            IF (Slot=9) THEN CALL Set_rela_d1_s9(Rel_num,State)
```

```
7840                 IF (Slot=10) THEN CALL Set_rela_d1_s10(Rel_num,State)
7850                 ELSE
7860                   CALL Drv_ps1c_rd_dvm
7870                 END IF
7880               END IF
7890            END IF
7900         END IF
7910  END IF
7920  ! Sw_state(Ipoint)=0
7930  !
7940  SUBEND ! UTILITY_SWITCH
7950  !
7951  !  ******************************************************************************
7952  !
7960  SUB Squib_driv                    !  PAR 68
7970  COM /Test_config/ Time_per_chan_1,Time_per_chan_2,Time_per_chan_3,Window_1
,Window_2,Window_3,Name$(20)[10],W1l1,W1l2,W2l1,W2l2,W2l3,W2l4,W3l1,W3l2
7980  COM /Test_config/ Chans_per_scn_1,Chans_per_scn_2,Chans_per_scn_3,Idle_tim
e_1,Idle_time_2,Start_chan,Stop_chan
7990  COM /Test_select/ Fil$(1:72)[80],Mark$[2],Test_name$[80]
8000  COM /Switch/ Sw_state(4),Srh,Sw_windows,Wait_gate(3)
8010     IF (Fil$(12)[3;2]="* ") THEN
8020     Test_name$=Fil$(12)[5;16]
8030  !
8040  !  SPECIFY THE Srh SWITCH USED TO START THE TEST.
8050  !  7 - EP & GUIDANCE       8 - SEEKER
8060  !
8070    Srh=7
8080  !
8090   CALL Init_pg_at  ! INITIALIZES PULSE GENS & ATTENUATORS BUT USED HERE
8100                    ! ONLY BECAUSE IT ALSO DISABLES THE ATTENUATOR SWITCH
8110                    ! USED IN THE Test_frame SUBROUTINE.
8120                    !
8140  FOR Ijk=1 TO 2    ! CAUSES THE TEST TO BE DONE MORE THAN ONCE.
8150  !
8160     !CALL Relay_card_chk
8170  !
8180  CALL Relay_init ! SETS RELAY OUTPUT CARDS TO PRE-TEST CONDITIONS.
8190  !
8200 CALL Set_rela_d1_s10(8,0)
8210   CALL Power_up("30S")   ! BRINGS UP POWER SUPPLIES ACCORDING TO SPEC.
8220  !
8230  !  SET APPROPRIATE RELAYS
8240  !
8250  IF (Ijk=1) THEN CALL Set_rela_d1_s8(3,0)
8260  !
8270  !CALL Reg_outpts ! CHECKS COPPERHEAD'S REGULATED OUTPUTS.
8280  !
8290  ! ESTABLISH WINDOW PARAMETERS.
8300  !
8310  Chans_per_scn_1=1
8320  Chans_per_scn_2=1
8330  Chans_per_scn_3=1
8340  Time_per_chan_1=1.E-4
8350  Time_per_chan_2=2.E-3
8360  Time_per_chan_3=5.E-4
8370  Window_1=3.5E-2
8380  Window_2=3.5
8390  Window_3=2.5
8400  !
8410  !  SET SCANNER CHANNEL POINTERS ACCORDING TO (SCANNER#-1)*32+CHANNEL#
8420  !
8430  W1l1=0
8440  W1l2=32
```

```
8450  !
8460  W2l1=1
8470  W2l2=33
8480  W2l3=65
8490  W2l4=97
8500  !
8510  W3l1=2
8520  W3l2=34
8530  !
8540  !  IDENTIFY CHANNEL ASSIGNMENTS IN MEANINGFUL TERMS.
8550  !
8560  Name$(0)="EA1_SQ_DR"
8570  Name$(1)="EA2_SQ_DR"
8580  Name$(2)="30V BAT"
8590  Name$(3)="SGG SQ DR"
8600  Name$(4)="GAS SQ DR"
8610  Name$(5)="SGS SQ DR"
8620  Name$(6)="WUN SQ DR"
8630  Name$(7)="WEX SQ DR"
8640  !
8650  !  EXECUTE THE TEST.
8660  !
8670  CALL Test_frame
8680  CALL Relay_init
8690 CALL Disab_sups_da !   Ftic_setup("DISABLE_SUPPLYS")
8700  !
8710  NEXT Ijk
8720     END IF
8730  Ftte_skip_test
8740  SUBEND
8750  !
8760  !     ************************************************************************
8770  !
8780  SUB Ad_ld_cal
8790  COM /Names/ Scanner,Cnt_tot,A_d1,A_d2,Multi_buffer,Timer_pacer,Timer_1,Rel
ay_d1_s7,Relay_d1_s8,Relay_d1_s9,Relay_d1_s10
8800  COM /Names/ A_d3,A_d4,D_a1,D_a2,D_a3,D_a4
8810  COM /Test_config/ Time_per_chan_1,Time_per_chan_2,Time_per_chan_3,Window_1
,Window_2,Window_3,Name$(20)[10],W1l1,W1l2,W2l1,W2l2,W2l3,W2l4,W3l1,W3l2
8820  COM /Test_config/ Chans_per_scn_1,Chans_per_scn_2,Chans_per_scn_3,Idle_tim
e_1,Idle_time_2,Start_chan,Stop_chan
8830  COM /Test_select/ Fil$(1:72)[80],Mark$[2],Test_name$[80]
8840  COM /Cal_fact/ Cal_gain(20,20),Cal_off(20,20),J_chan_pt,J_chan_beg,Dvm_vol
ts(100),Ad_volts(110),Load_gain(130),Ircnt,Xoff_fit,Yoff_fit
8850  COM /Switch/ Sw_state(4),Srh,Sw_windows,Wait_gate(3)
8860 !DIM Gain(4,4)
8870  DIM Vlts(700)
8880     IF (Fil$(23)[3;2]="* ") THEN
8890     Test_name$=Fil$(23)[5;16]
8900  !
8910  !
8920  !     THIS ROUTINE DRIVES THE FOUR (SCANNER-OPAMP-A/D) LINES WITHOUT
8930  ! LOADS TO DETERMINE OPAMP-A/D GAIN AND OFFSET PARAMETERS.  THESE
8940  ! ARE THEN USED TO OBTAIN THE TRUE VALUES OF ALL OTHER SCANNER VOLTAGE
8950  ! DIVIDERS.  DIVIDER (LOAD_GAIN) VALUES AND OPAMP-A/D PARAMETERS ARE
8960  ! THEN STORED FOR LATER USE DURING ANALYSIS.
8970  !
8980   CALL Init_pg_at  ! INITIALIZES PULSE GENS & ATTENUATORS BUT USED HERE
8990                    ! ONLY BECAUSE IT ALSO DISABLES THE ATTENUATOR SWITCH
9000                    ! USED IN THE Test_frame SUBROUTINE
9010                    !
9020  !
9030  ! TURN OFF WAITS OF SECOND WINDOW
9040  !
```

```
9050  Wait_gate(1)=0
9060  Wait_gate(2)=0
9070  !
9080  Srh=0
9140  FOR Ijk=1 TO 1   ! CAUSES THE TEST TO BE DONE MORE THAN ONCE.
9150  !
9160  CALL Relay_init ! SETS RELAY OUTPUT CARDS TO PRE-TEST CONDITIONS.
9170  !
9180  WAIT 1
9190 CALL Set_rela_d1_s10(8,0)
9200 ! CALL Power_up("30")   ! BRINGS UP POWER SUPPLIES ACCORDING TO SPEC.
9210  !
9220  !  SET APPROPRIATE RELAYS
9230  !
9240    CALL Set_rela_d1_s9(5,0)
9250  !
9260  ! SET 2P2-5 GROUND
9270  !
9280  ! CALL Set_rela_d1_s9(8,0)
9290  ! CALL Set_rela_d1_s9(11,0)
9300  ! CALL Set_rela_d1_s9(14,0)
9310  ! CALL Set_rela_d1_s8(1,0)
9320  !
9330  ! CONNECT GUIDANCE LOADS
9340  !
9350    CALL Set_rela_d1_s10(8,0)     ! REMOVES HIGH CURRENT SEEKER LOADS
9360                                  ! FROM THE CALIBRATE BUS
9370    CALL Set_rela_d1_s8(15,1)     !  DISCONNECTS HIGH CURRENT RESISTOR PLATE
9380                                  !  LOADS FROM CAL BUS
9390  !
9400  Ftic_setup("INIT_DVM")     ! INITIALIZE THE DVM
9410  !
9420     ! CLOSE CHANNEL 7
9430     !
9440     Ftic_setup("CONNECT_DVM","7")
9450     !
9460  !  SET OUTSIDE ACCESS RELAYS(TO CHANNEL 7 OF DVM)
9470  !
9480  CALL Set_rela_d1_s10(13,1)
9490  CALL Set_rela_d1_s10(14,1)
9500  !
9510     ! SET D/A VOLTAGES TO ZERO OR 5 VOLTS
9520     !
9530     Volts=0
9540     IF (Ijk=2) THEN Volts=5
9550     Preset(D_a1,Volts)
9560     Preset(D_a2,Volts)
9570     Preset(D_a3,Volts)
9580     Preset(D_a4,Volts)
9590     Start(D_a1)
9600     Start(D_a2)
9610     Start(D_a3)
9620     Start(D_a4)
9630  !
9640  ! SET PS1B TO ZERO
9650  !
9660     Ftic_setup("INIT_PS1B","0",".75")
9670  !
9680  !  CONNECT THE DVM TO THE POWER SUPPLY
9690  !
9700  !  CONNECT THE POWER SUPPLY TO ALL OF THE COPPERHEAD OUTPUT ATTENUATORS
9710  !  IN SUCH A WAY AS TO APPLY A NEGATIVE VOLTAGE
9720  !
9730     OUTPUT 709;"CHAN?"
```

```
9740      ENTER 709;A
9750      PRINT "CHANNEL ",A," IS CONNECTED AND THE D/As ARE SET TO ",Volts," VOL
TS."
9760      !
9770  ! SET PS1B TO NEGATIVE POLARITY
9780  !
9790      CALL Set_rela_d1_s10(5,0)
9800      CALL Set_rela_d1_s10(1,0)
9810      CALL Set_rela_d1_s10(2,0)
9820      CALL Set_rela_d1_s10(3,0)
9830      WAIT 2.0E-2
9840  !
9850   PRINT "APPLY ONLY THE DC SUPPLY TO THE LOADS AT THIS TIME."
9860   PRINT
9870   PRINT "TYPE ENTER TO CONTINUE."
9880   INPUT Anss$
9890   PRINT
9900   PRINT "SET THE FOLLOWING CALIBRATION SWITCHES TO THE ""O"" POSITION."
9910   PRINT "ALL OTHER SWITCHES SHOULD BE IN THE ""G"" POSITION."
9920   PRINT "AND TYPE ENTER TO CONTINUE."
9930   PRINT
9940   PRINT "P2-2,P3-2,P2-4,P3-4,ALL"
9950   PRINT
9960   INPUT Anss$
9970      Ftic_setup("SET_PS1B_VOLTAG","9")
9980      CALL Read_dvm(Dvm_no,7,Units$,Reading)
9990      WAIT 2
10000   ! CALL Read_dvm(Dvm_no,1,Units$,Readb)
10010   ! Reading=Reading-Readb
10020     Dvm_volts(1)=Reading
10030! CALL Reg_outpts ! CHECKS COPPERHEAD'S REGULATED OUTPUTS.
10040 !
10050 ! ESTABLISH WINDOW PARAMETERS.
10060 !
10070 Chans_per_scn_1=1
10080 Chans_per_scn_2=1
10090 Chans_per_scn_3=1
10100 Time_per_chan_1=1.E-4           !1E-4
10110 Time_per_chan_2=1.5E-2  !20               !2E-3
10120 Time_per_chan_3=1.E-4
10130 Window_1=1.E-4                 !12E-3
10140 Window_2=63                   !3.5
10150 Window_3=1.E-4              !2.5
10151 !
10160 Sw_windows=10        ! SAYS THAT THE UTILITY SWITCH WILL BE USED IN
10161                      ! WINDOW 2
10162                      ! 100 FOR WINDOW 1 , 10 FOR 2 , AND 1 FOR 3.
10163 Sw_state(2)=1000000   !IDENTIFIES A RUN AS BEING FOR THE PURPOSE OF
10164 !                      CALIBRATION.
10166 !
10167 !  SET SCANNER CHANNEL POINTERS ACCORDING TO (SCANNER#-1)*32+CHANNEL#
10168 !
10169 W1l1=0
10170 W1l2=32
10171 !
10172 W2l1=7
10173 W2l2=44
10174 W2l3=68
10175 W2l4=103
10176 !
10177 W3l1=2
10178 W3l2=34
10179 !
10180 !  IDENTIFY CHANNEL ASSIGNMENTS IN MEANINGFUL TERMS.
```

```
10181 !
10182 Name$(0)="EA1_SQ_DR"
10183 Name$(1)="EA2_SQ_DR"
10184 Name$(2)="SC/OP/AD1"
10185 Name$(3)="SC/OP/AD2"
10186 Name$(4)="SC/OP/AD3"
10187 Name$(5)="SC/OP/AD4"
10188 Name$(6)="WUN SQ DR"
10189 Name$(7)="WEX SQ DR"
10190 !
10191 !  EXECUTE THE TEST.
10192 !
10193   CALL Test_frame
10194 CALL Disab_sups_da !   Ftic_setup("DISABLE_SUPPLYS")
10195 !
10196 NEXT Ijk
10197 !
10198 ! A_D AND OP AMP GAINS AND OFFSETS COMPLETED.  NOW DETERMINE
10199 ! ATTENUATOR VALUES FROM THE FOLLOWING MEASUREMENTS.
10200 !
10201 ! Disable_mxctrl(A_d1)
10202 ! Disable_mxctrl(A_d2)
10203 ! Disable_mxctrl(A_d3)
10204 ! Disable_mxctrl(A_d4)
10205   Set_switches(Scanner,0)
10206   GOTO Jmp_over
10207   Ftic_setup("SET_PS1B_VOLTAG","9")
10208    DIM Drop_fact(5)
10209    FOR Izi=1 TO 5
10210    CALL Ground_select(Izi)
10211    WAIT 2.0E-2
10212    CALL Read_dvm(1,7,Unit$,Reading_in)
10213    WAIT 3
10214    CALL Read_dvm(1,1,Unit$,Reading_drop)
10215    Drop_fact(Izi)=Reading_drop/Reading_in
10216    NEXT Izi
10217 Jmp_over:   !
10218   Ftic_setup("SET_PS1B_VOLTAG","0")
10219   PRINT "REMOVE THE DC SUPPLY FROM THE LOADS AT THIS TIME AND"
10220   PRINT "APPLY ONLY THE PULSE GENERATOR TO THE LOADS."
10221   PRINT
10222   PRINT "TYPE ENTER TO CONTINUE."
10223   INPUT Anss$
10224 ! Ftic_setup("SET_PS1B_VOLTAG","1.5")
10225 ! Ftic_setup("SET_PS1B_VOLTAG","28")
10226 !
10227 ! CHOOSE THE TYPE OF CALIBRATION TO BE PERFORMED. (SEEKER OR EP)
10228 !
10229 Bad:!
10230 PRINT
10231 PRINT "DO YOU WISH TO PERFORM AN EP(E) ¯ . A SEEKER(S) CALIBRATION ?"
10232 INPUT Anss$
10233 IF (Anss$="E") THEN
10234   Cal_type=1
10235   CALL Set_rela_d1_s10(8,0)    ! APPLIES HIGH CURRENT EP LOADS
10236  ELSE
10237   IF (Anss$<>"S") THEN
10238   PRINT "ERROR -- AN IMPROPER TEST CHARACHTER HAS BEEN CHOSEN."
10239   PRINT "TYPE ENTER TO TRY AGAIN." .
10240   GOTO Bad
10241   END IF
10242   Cal_type=2
10243 END IF
10244   PRINTER IS 9
```

```
10245 !
10246 !
10247 ! DEFINE THE INPUT VOLTAGE PULSE HEIGHTS, GROUND SELECTIONS,
10248 ! EXPECTED LOADS ASSOCIATED WITH EACH CHANNEL, AND THE PIN NUMBER
10249 ! FOR EACH CHANNEL.
10250 !
10251   DIM Chan_sup(127),Chan_gnd(127),Chan_load(127),Chan_pin$(127)[8]
10252   DIM Pulse_amp(6),Chan_gnd$(6)[8],Ret_drop(127),Chan_ret(127)
10253   !
10254   !   CASE NUMBER               PULSE HEIGHT
10255   !
10256   !       2                        11 VOLTS
10257   !       3                        13 VOLTS
10258   !       4                        24 VOLTS
10259   !       5                        29 VOLTS
10260   !       6                         9 VOLTS
10261   !
10262   DATA 2,2,5,3,6,5,6,6,6,5,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
10263   DATA 2,5,5,6,6,2,6,5,6,5,3,6,6,3,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
10264   DATA 6,5,3,5,6,6,6,6,5,5,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
10265   DATA 6,5,6,5,3,6,6,6,5,3,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
10266   READ Chan_sup(*)
10267   !
10268   !     CASE                     GROUND SELECTION
10269   !
10270   !      1                         2P3-18
10271   !      2                         2P3-17
10272   !      3                         2W1A-6
10273   !      4                         2W1D-6
10274   !      5                         2P2-5
10275   !      6                         2P3-5
10276   !
10277   Chan_gnd$(1)="2P3-18"
10278   Chan_gnd$(2)="2P3-17"
10279   Chan_gnd$(3)="2W1A-6"
10280   Chan_gnd$(4)="2W1D-6"
10281   Chan_gnd$(5)="2P2-5"
10282   Chan_gnd$(6)="2P3_5"
10283   !
10284   DATA 1,1,1,2,4,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10285   DATA 1,1,1,2,4,2,2,2,3,3,3,2,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10286   DATA 1,1,3,5,2,1,2,2,3,3,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10287   DATA 1,1,1,5,2,1,1,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10288   READ Chan_gnd(*)
10289   !
10290   DATA 6,1,1,2,4,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10291   DATA 5,1,1,2,4,2,2,1,3,3,3,2,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10292   DATA 1,1,3,6,2,1,2,2,3,3,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10293   DATA 1,1,1,5,2,1,1,2,3,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
10294   READ Chan_ret(*)
10295   !
10296   !  EXPECTED LOAD VALUES
10297   !
10298   DATA 1E3,1E3,1E3,1.422E6,1E20,1.2E6,1E20,1E20,1E20,7.5E3,1E20,1E20
10299   DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10300   DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10301   !
10302   DATA 1E3,1E3,1E3,1E20,1E20,149.2E3,1E20,1E3,1E20,1E20,1.511E6,1E20
10303   DATA 1E20,1.511E6,1E20,1E20
10304   DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10305   DATA 1E20,1E20,1E20,1E20
10306   !
10307   DATA 1E20,1E3,1.511E6,10E3,1E20,1E20,1E20,1E20,10E3,10E3,1E20
10308   DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
```

```
10309     DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10310     !
10311     DATA 1E20,1E3,1E20,10E3,1.422E6,1E20,1E20,1E20,10E3,1.511E6
10312     DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10313     DATA 1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20,1E20
10314     !
10315     READ Chan_load(*)
10316     !
10317     DATA "2P3-2","2P3-20","2P2-12","2W1C-4","2W1D-17","2P3-22","2P2-7"
10318     DATA "2P3-21","2W1A-11","2W1A-3"
10319     DATA "","","","","","","","","","","","","","","","","","","",""
10320     DATA "",""
10321     !
10322     DATA "2P2-2","2P2-25","2P2--15","2W1D-19","2P2-21","2P3-10","2W1D-21"
10323     DATA "2P2-15","2W1A-12","2W1A-3","2W1A-4","2P3-9","2P3-21","2W1A-5"
10324     DATA "2W1A-7"
10325     DATA "","","","","","","","","","","","","","","","",""
10326     !
10327     DATA "","2P2-19","2W1A-5","2P3-4","2P2-22","","2W1D-7","2P3-8","2W1A-8"
10328     DATA "2W1A-2","","2W1C-12"
10329     DATA "","","","","","","","","","","","","","","","","","","",""
10330     !
10331     DATA "","2P3-25","","2P2-4","2P3-6","","","2P3-9","2W1A-10","2W1A-5"
10332     DATA "","","","","","","","","","","","","","","","","","","",""
10333     DATA "",""
10334     READ Chan_pin$(*)
10335     !
10336     DATA 0,9,11,13,24,29,9
10337     READ Pulse_amp(*)
10338     !
10339 !
10340      FOR Sup_indx=2 TO 6
10341      PRINTER IS 1
10342       Set_switches(Scanner,0)
10343       PRINT "SET THE PULSE GENERATOR'S TRIGGER SWITCH TO MANUAL AND"
10344       SELECT Sup_indx
10345       CASE 2
10346       PRINT "SET THE INPUT'S PULSE HEIGHT TO 11 VOLTS AND TYPE ENTER"
10347       PRINT "TO CONTINUE."
10348       INPUT Anss$
10349       CASE 3
10350       PRINT "SET THE INPUT'S PULSE HEIGHT TO 13 VOLTS AND TYPE ENTER"
10351       PRINT "TO CONTINUE."
10352       INPUT Anss$
10353       CASE 4
10354       PRINT "SET THE INPUT'S PULSE HEIGHT TO 24 VOLTS AND TYPE ENTER"
10355       PRINT "TO CONTINUE."
10356       INPUT Anss$
10357       CASE 5
10358       PRINT "SET THE INPUT'S PULSE HEIGHT TO 29 VOLTS AND TYPE ENTER"
10359       PRINT "TO CONTINUE."
10360       INPUT Anss$
10361       CASE 6
10362       PRINT "SET THE INPUT'S PULSE HEIGHT TO 9 VOLTS AND TYPE ENTER"
10363       PRINT "TO CONTINUE."
10364       INPUT Anss$
10365       END SELECT
10366       PRINT
10367       PRINT "REMOVE THE SCOPE FROM THE SYSTEM."
10368       PRINT "TYPE ENTER TO CONTINUE."
10369       INPUT Anss$
10370       PRINT
10371       PRINT "SET THE PULSE GENERATOR'S TRIGGER SWITCH TO EXTERNAL."
10372       PRINT "TYPE ENTER TO CONTINUE."
```

```
10373      INPUT Anss$
10374     FOR Gnd_indx=1 TO 5
10375     CALL Ground_select(Gnd_indx)
10376    FOR Ijk=0 TO 127      !  67 TO 68
10377    IF (Cal_flag=1) THEN
10378     PRINT "RETURN SWITCH TO ""G"" POSITION AND TYPE ENTER TO CONTINUE. "
10379     INPUT Anss$
10380     Cal_flag=0
10381    END IF
10382 ! Disable_extrig(A_d1)
10383 ! Disable_extrig(A_d2)
10384 ! Disable_extrig(A_d3)
10385 ! Disable_extrig(A_d4)
10386 ! Ik=Ijk-1
10387    Ik=Ijk
10388    Set_switches(Scanner,0)
10389    ! OPEN TIMER TO SCANNER RELAY HERE
10390    CALL Set_rela_d1_s7(10,0)
10391    Disable_extrig(Scanner)
10392    WAIT 2.0E-2
10393    Fit_pointer=3    !    3 LINEAR , 4  -  A*EXP(BX+CX2)
10394    SELECT Ik
10395    CASE <32
10396    IF (Chan_sup(Ik)<>Sup_indx) THEN GOTO Next_ijk
10397    IF (Chan_gnd(Ik)<>Gnd_indx) THEN GOTO Next_ijk
10398    GOSUB Cal_switch
10399 ! GOTO NEXT_IJK
10400 ! Enable_extrig(A_d1)
10401    Write_start_chn(Scanner,Ik)
10402    WAIT .1
10403    Set_switches(Scanner,1)
10404    WAIT .1
10405    Ad_no=1
10406    GOSUB Get_data
10407    SELECT Fit_pointer
10408    CASE =3
10409!
10410! USED FOR A LINEAR FIT
10411    Load_gain(Ijk)=(Reading_b-Cal_off(2,1))/(Cal_gain(2,1)*(Reading_a-Readin
g_c))
10412!
10413!
10414    CASE =4
10415    Aeq9=Cal_gain(3,0)
10416    Beq9=Cal_gain(3,1)
10417    Ceq9=Cal_gain(3,2)
10418    Const1=-LOG((Reading_b+Yoff_fit)/Aeq9)
10419    Vin_quad_1=(-Beq9+SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10420    Vin_quad_2=(-Beq9-SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10421    Load_gain(Ijk)=(MAX(Vin_quad_1,Vin_quad_2)-Xoff_fit)/Reading_a
10422    END SELECT
10423    CASE 32 TO 63
10424     IF (Chan_sup(Ik)<>Sup_indx) THEN GOTO Next_ijk
10425     IF (Chan_gnd(Ik)<>Gnd_indx) THEN GOTO Next_ijk
10426     GOSUB Cal_switch
10427 ! IF (Ik<>32) THEN GOTO Next_ijk
10428 ! PRINT "SET CALIBRATE VOLTAGE TO 11.5 VOLTS."
10429 ! INPUT Anss$
10430 ! Enable_extrig(A_d2)
10431    Write_start_chn(Scanner,Ik)
10432    Set_switches(Scanner,1)
10433 ! CALL Read_dvm(1,7,Unit$,Reading_a)
10434 ! Tot_read=0
10435 !   FOR Lp=1 TO 20
```

```
10436 !  Start(A_d2)
10437 !  Input(A_d2,Reading_b)
10438 !  Tot_read=Tot_read+Reading_b
10439 !  NEXT Lp
10440 ! Reading_b=-Tot_read/20
10441   Ad_no=2
10442 GOSUB Get_data
10443   SELECT Fit_pointer
10444   CASE =3
10445!
10446! USED FOR A LINEAR FIT
10447   Load_gain(Ijk)=(Reading_b-Cal_off(2,2))/(Cal_gain(2,2)*(Reading_a-Readin
g_c))
10448!
10449!
10450   CASE =4
10451   Aeq9=Cal_gain(4,0)
10452   Beq9=Cal_gain(4,1)
10453   Ceq9=Cal_gain(4,2)
10454   Const1=-LOG((Reading_b+Yoff_fit)/Aeq9)
10455   Vin_quad_1=(-Beq9+SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10456   Vin_quad_2=(-Beq9-SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10457   Load_gain(Ijk)=(MAX(Vin_quad_1,Vin_quad_2)-Xoff_fit)/Reading_a
10458   END SELECT
10459   CASE 64 TO 95
10460   IF (Chan_sup(Ik)<>Sup_indx) THEN GOTO Next_ijk
10461   IF (Chan_gnd(Ik)<>Gnd_indx) THEN GOTO Next_ijk
10462   GOSUB Cal_switch
10463 ! GOTO Next_ijk
10464 ! Enable_extrig(A_d3)
10465 ! IF (Ik=68) THEN
10466 ! Ftic_setup("SET_PS1B_VOLTAG","9")
10467 ! END IF
10468   Write_start_chn(Scanner,Ik)
10469   Set_switches(Scanner,1)
10470!  CALL Read_dvm(1,7,Unit$,Reading_a)
10471   Tot_read=0
10472!  Zz=0
10473!  Tot_load=0
10474 Zz_chk:  !
10475!    Zz=Zz+1
10476!    IF (Zz=8) THEN GOTO Jmp_here
10477!    Vol_valz=.1*Zz+.8
10478!    Vol_valz$=VAL$(Vol_valz)
10479!  Ftic_setup("SET_PS1B_VOLTAG",Vol_valz$)
10480!  WAIT 1
10481!  CALL Read_dvm(1,7,Unit$,Reading_a)
10482!   FOR Lp=1 TO 20
10483!   Start(A_d3)
10484!   Input(A_d3,Reading_b)
10485!   Tot_read=Tot_read+Reading_b
10486!   NEXT Lp
10487!  Reading_b=-Tot_read/20
10488   Ad_no=3
10489 GOSUB Get_data
10490   SELECT Fit_pointer
10491   CASE =3
10492!
10493! USED FOR A LINEAR FIT
10494   Load_gain(Ijk)=(Reading_b-Cal_off(4,1))/(Cal_gain(4,1)*(Reading_a-Readin
g_c))
10495!  Tot_load=Load_gain(Ijk)+Tot_load
10496!  GOTO Zz_chk
10497 Jmp_here:  !
```

```
10498!  Load_gain(Ijk)=Tot_load/7
10499!
10500!
10501   CASE =4
10502   Aeq9=Cal_gain(5,0)
10503   Beq9=Cal_gain(5,1)
10504   Ceq9=Cal_gain(5,2)
10505   Const1=-LOG((Reading_b+Yoff_fit)/Aeq9)
10506   Vin_quad_1=(-Beq9+SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10507   Vin_quad_2=(-Beq9-SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10508   Load_gain(Ijk)=(MAX(Vin_quad_1,Vin_quad_2)-Xoff_fit)/Reading_a
10509   Load_gain(Ijk)=(Reading_b+Cal_gain(4,1)*Cal_off(2,2))/(Cal_gain(4,1)*Rea
ding_a)
10510   END SELECT
10511   CASE >95
10512   IF (Chan_sup(Ik)<>Sup_indx) THEN GOTO Next_ijk
10513   IF (Chan_gnd(Ik)<>Gnd_indx) THEN GOTO Next_ijk
10514   GOSUB Cal_switch
10515!  IF (Ik<>99) THEN GOTO Next_ijk
10516!  PRINT "SET CALIBRATE VOLTAGE TO 27 VOLTS."
10517 ! INPUT Anss$
10518 ! Enable_extrig(A_d4)
10519   Write_start_chn(Scanner,Ik)
10520   Set_switches(Scanner,1)
10521!  CALL Read_dvm(1,7,Unit$,Reading_a)
10522!  Tot_read=0
10523!   FOR Lp=1 TO 20
10524!   Start(A_d4)
10525!   Input(A_d4,Reading_b)
10526!   Tot_read=Tot_read+Reading_b
10527!   NEXT Lp
10528!  Reading_b=-Tot_read/20
10529   Ad_no=4
10530  GOSUB Get_data
10531   SELECT Fit_pointer
10532   CASE =3
10533!
10534! USED FOR A LINEAR FIT
10535   Load_gain(Ijk)=(Reading_b-Cal_off(4,2))/(Cal_gain(4,2)*(Reading_a-Readin
g_c))
10536!
10537!
10538   CASE =4
10539   Aeq9=Cal_gain(6,0)
10540   Beq9=Cal_gain(6,1)
10541   Ceq9=Cal_gain(6,2)
10542   Const1=-LOG((Reading_b+Yoff_fit)/Aeq9)
10543   Vin_quad_1=(-Beq9+SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10544   Vin_quad_2=(-Beq9-SQRT(Beq9*Beq9-4*Const1*Ceq9))/(2*Ceq9)
10545   Load_gain(Ijk)=(MAX(Vin_quad_1,Vin_quad_2)-Xoff_fit)/Reading_a
10546   Load_gain(Ijk)=(Reading_b+Cal_gain(4,2)*Cal_off(4,1))/(Cal_gain(4,2)*Rea
ding_a)
10547   END SELECT
10548 ! CASE ELSE
10549 ! Load_gain(Ijk)=1
10550   END SELECT
10551   PRINT "     CHANNEL NUMBER              LOAD GAIN"
10552   PRINT
10553   PRINT "      ",Ijk,Load_gain(Ijk)
10554   PRINT
10555 Next_ijk:  !
10556    NEXT Ijk
10557     NEXT Gnd_indx
10558      NEXT Sup_indx
```

```
10559   Set_switches(Scanner,0)
10560   PRINT
10561   GOTO Leap
10562      FOR Kch=0 TO 128
10563      PRINT "CHANNEL NUMBER = ",Kch
10564      PRINT "OLD LOAD_GAIN = ",Load_gain(Kch)
10565      SELECT Kch
10566      CASE <32
10567      Load_gain(Kch)=Load_gain(Kch)+(1-Load_gain(7))
10568      CASE 32 TO 63
10569      Load_gain(Kch)=Load_gain(Kch)+(1-Load_gain(44))
10570      CASE 64 TO 95
10571      Load_gain(Kch)=Load_gain(Kch)+(1-Load_gain(68))
10572      CASE >95
10573      Load_gain(Kch)=Load_gain(Kch)+(1-Load_gain(103))
10574      END SELECT
10575      PRINT
10576      PRINT "NEW LOAD_GAIN = ",Load_gain(Kch)
10577      PRINT
10578      NEXT Kch
10579 Leap:  !
10580 ! CALCULATE A CALIBRATION FACTOR
10581 !
10582!PRINT
10583!PRINT "OFFSET   CAL_FACTOR   AVERAGE VOLTS OUT    DVM IN    OFFSET"
10584!PRINT
10585! FOR I=0 TO 6 STEP 2
10586! FOR J=0 TO 1
10587! Cal_fact(I,J)=Avg_volts(I,J)/(Reading-Offset)
10588! PRINT Offset,Cal_fact(I,J),Avg_volts(I,J),Reading,Offset
10589! NEXT J
10590! NEXT I
10591  !
10592  ! PRINT CALIBRATION DATA
10593  PRINTER IS 9
10594  !
10595  PRINT "   CAL_GAIN_1          CAL_GAIN_2        CAL_GAIN_3       CAL_GAIN_4
"
10596  PRINT
10597  PRINT Cal_gain(2,1),Cal_gain(2,2),Cal_gain(4,1),Cal_gain(4,2)
10598  PRINT
10599  PRINT "   CAL_OFF_1           CAL_OFF__2        CAL_OFF_3        CAL_OFF_4"
10600  PRINT
10601  PRINT Cal_off(2,1),Cal_off(2,2),Cal_off(4,1),Cal_off(4,2)
10602   INTEGER Ic_prt
10603   FOR Ic=0 TO 127
10604   IF (Ic MOD 32=0) THEN
10605   PRINT
10606   OUTPUT 9;CHR$(12)
10607  IF (Ic=0) THEN PRINT USING Format
10608 Format: IMAGE 4/,26X,"LOAD CALIBRATION REPORT"
10609   PRINT
10610   PRINT
10611   PRINT
10612   PRINT "SCANNER ",INT(Ic/32)+1
10613   PRINT
10614  PRINT "CHAN      PIN        WRT GND   RET GND  DVD LOAD  DVD GAIN  CAL VL
TG  RET DROP"
10615  PRINT
10616   END IF
10617   Load_gain_prt=INT(Load_gain(Ic)*1.E+5)/1.E+5
10618   Ic_prt=Ic
10619  PRINT Ic_prt,Chan_pin$(Ic),Chan_gnd$(Chan_gnd(Ic)),Chan_gnd$(Chan_ret(Ic)
),Chan_load(Ic),Load_gain_prt,Pulse_amp(Chan_sup(Ic)),Ret_drop(Ic)
```

```
10620   NEXT Ic
10621   !
10622  ! STORE CALIBRATION DATA
10623  !
10624  PRINT
10625  PURGE "/FTM/HOME_DIR00/CAL_FACT:,1400"
10626  CREATE ASCII "/FTM/HOME_DIR00/CAL_FACT:,1400",200
10627  ASSIGN @File TO "/FTM/HOME_DIR00/CAL_FACT:,1400"
10628  OUTPUT @File;Cal_gain(*),Cal_off(*),Load_gain(*)
10629  ! END IF
10630    PRINTER IS 1
10631    PRINT "REMOVE THE PULSE GENERATOR NOW !!!"
10632    PRINT
10633    PRINT "TYPE ENTER TO CONTINUE."
10634    INPUT Anss$
10635    CALL Relay_init
10636 !
10637   CALL Set_rela_d1_s8(15,0)     ! DISCONNECTS HIGH CURRENT SEEKER LOAD
10638                                 ! DIVIDERS
10639 !
10640 Ftte_skip_test
10641 Get_data:!
10642          PRINTER IS 1
10643   !
10644   Tot_read=0
10645  !Tot_read_tot=0
10646    FOR Lp=1 TO 1   ! 2
10647 !  Adjust:   !
10648    Tot_read=0
10649   !
10650   ! SET UP DVM TO TRIGGER ON ITS EXTERNAL INPUT, DELAY FOR 2E-3 SECONDS
10651   ! AND GET A DCV (AVERAGE) OVER AN INTEGRATION TIME OF 1.7E-3 SECONDS.
10652   !
10653 CLEAR 709
10654 OUTPUT 709;"RESET"
10655 !  OUTPUT 709;"DCV 30,.01"
10656 !  WAIT 5
10657 !  IF (Lp=2) THEN
10658 !  OUTPUT 709;"DCV .29"
10659 !  ELSE
10660    OUTPUT 709;"DCV 30"
10661 !  END IF
10662    OUTPUT 709 USING "#,K";"TARM HOLD;"
10663    OUTPUT 709;"TERM,2"
10664 !  OUTPUT 709;"TERM?"
10665 !  ENTER 709;Ans
10666 !  PRINT "TERM=",Ans
10667    OUTPUT 709;"CHAN,7"
10668 !  OUTPUT 709;"CHAN?"
10669 !  ENTER 709;Ans
10670 !  PRINT "CHAN=",Ans
10671 !  IF (Lp=2) THEN OUTPUT 709;"CHAN,1"
10672    OUTPUT 709;"DELAY 2E-3"
10673    OUTPUT 709;"NPLC .1"
10674    OUTPUT 709 USING "#,K";"TARM 1;"
10675    OUTPUT 709 USING "#,K";"TRIG EXT;"
10676    OUTPUT 709;"NRDGS 1,2"
10677 !  OUTPUT 709 USING "#,K";"TARM SGL,0;"
10678 Init(Multi_buffer)
10679 Init(A_d1)
10680 Init(A_d2)
10681 Init(A_d3)
10682 Init(A_d4)
10683 !  Init(Multi_buffer)
```

```
10684     Set_range(A_d1,10)
10685     Set_range(A_d2,10)
10686     Set_range(A_d3,10)
10687     Set_range(A_d4,10)
10688   Write_dif_count(Multi_buffer,0)
10689   Write_wpointer(Multi_buffer,0)
10690   Write_rpointer(Multi_buffer,0)
10691   Set_ext_pacing(Scanner)
10692   Disable_extrig(Scanner)
10693   Set_crosspoint(Scanner)
10694   Triggers=65536-366
10695   Output(Cnt_tot,Triggers)
10696   Set_period(Scanner,1.0E-5)
10697   !PRINT "START SCANNER."
10698   !INPUT Anss$
10699 ! WAIT 2
10700 ! OUTPUT 710;"VSET 3,0"
10701 ! WAIT 2
10702 WAIT 1
10703   OUTPUT 710;"VSET 3,"&VAL$(Pulse_amp(Sup_indx))
10704   WAIT .02
10705   Start(Scanner)
10706 Spin:  !
10707   Check_done(Cnt_tot,Cnt_flag)
10708   IF (Cnt_flag<>1) THEN
10709   GOTO Spin
10710   ELSE
10711   Enable_lockout(Multi_buffer)
10712   OUTPUT 710;"VSET 3,0"
10713   WAIT 1
10714   PRINT "DATA COLLECTED"
10715   END IF
10716   Read_wpointer(Multi_buffer,Ans)
10717   PRINT Ans
10718   Set_fifo_in(Multi_buffer)
10719   Write_rpointer(Multi_buffer,796)
10720 !  IF (Lp=10) THEN
10721   CONTROL 32,1;0
10722   Input_rblock(Multi_buffer,Vlts(*),665)
10723   CONTROL 32,1;1
10724 !  END IF
10725 Icnt=0
10726 Tot_read=0
10727   FOR Ixd=1 TO 165
10728   Icnt=Icnt+1
10729   Izd=(Ixd-1)*4+(Ad_no-1)
10730   Vlts(Izd)=-1*Vlts(Izd)*5.0E-3
10731   Tot_read=Vlts(Izd)+Tot_read
10732   NEXT Ixd
10733   Tot_read=Tot_read/Icnt
10734 ! Duty=.1
10735 ! Tot_read=Tot_read/Duty
10736 ! Tot_read_tot=Tot_read_tot+Tot_read
10737 !    PRINT "TRY AND GET DATA"
10738 !    INPUT Anss$
10739 ! IF (Lp=1) THEN
10740   Reading_b=Tot_read
10741   ENTER 709;Reading_a
10742   PRINT "AVERAGE INPUT VOLTAGE  = ",Reading_a
10743   PRINT "AVERAGE OUTPUT VOLTAGE = ",Reading_b
10744 ! PRINT "TURN OFF 30V PULSE GENERATOR."
10745 ! INPUT Anss$
10746 ! OUTPUT 709;"CHAN,1"
10747 ! OUTPUT 709;"TRIG SGL"
```

```
10748 !    PRINT "TYPE ""G"" FOLLOWED BY A RETURN TO RESUME CALIBRATION."
10749  !    PRINT
10750  !    PRINT "TYPE RETURN BY ITSELF TO ADJUST THE INPUT PULSE LEVEL."
10751  !    INPUT Anss$
10752  !    IF (Anss$<>"G") THEN GOTO Adjust
10753 ! ELSE
10754  !  ENTER 709;R_ed_c
10755   !
10756   ! IF COPPERHEAD'S MEASUREMENT IS MADE WITH RESPECT TO A GROUND WHICH
10757   ! IS DIFFERENT FROM THE LOAD'S RETURN, DO NOT REPLACE THE RETURN LINE
10758   ! VOLTAGE DROP LOST WHILE ELIMINATING ERRONIOUS I*R DROPS (READING_C)
10759   ! GENERATED BY CURRENTS SHARING THE SAME RETURN LINE DURING CALIBRATION.
10760   !
10761   GOTO Leap_1
10762     R_ed_c=Drop_fact(Gnd_indx)*Reading_a
10763     Reading_c=R_ed_c
10764     SELECT Ik .
10765     CASE 0,32,39,67
10766     GOTO No_drop
10767     CASE ELSE
10768     Line_cur=Reading_a/Chan_load(Ik)
10769     Reading_c=R_ed_c-.25*Line_cur
10770     END SELECT
10771 No_drop:     !
10772   PRINT "AVERAGE LINE DROP = ",R_ed_c
10773   Ret_drop(Ik)=R_ed_c
10774 Leap_1:  !
10775   Reading_c=0
10776 ! END IF
10777 ! PRINT "TYPE ENTER TO GO AGAIN."
10778 ! INPUT Anss$
10779    NEXT Lp
10780    RETURN
10781 Cal_switch:   !
10782   SELECT Ik
10783    CASE 1,2,33,34,39,65,97
10784     PRINT
10785     PRINT "CLOSE ONLY SWITCH "&Chan_pin$(Ik)&"."
10786     PRINT
10787     PRINT "TYPE ENTER TO CONTINUE."
10788     INPUT Anss$
10789     Cal_flag=1
10790    CASE ELSE
10791    END SELECT
10792 RETURN
10793    END IF
10794 Ftte_skip_test
10795 SUBEND  ! AD_LD_CAL
10796 !
10797 !   ******************************************************************
10799 !
10800 SUB Drv_ps1c_rd_dvm
10801 !
10802 ! THIS ROUTINE IS USED TO DRIVE PS1B FROM -10 TO +10 VOLTS IN
10803 ! 1 VOLT STEPS WHILE READING DVM VALUES.  THE VALUES ARE COLLECTED
10804 ! ALONG WITH A/D READINGS TAKEN SIMULTANEOUSLY BY THE 6944 FOR THE
10805 ! PURPOSE OF CALIBRATING CHANNEL ATTENUATORS AND OP/AMPS.
10806 !
10807 COM /Names/ Scanner,Cnt_tot,A_d1,A_d2,Multi_buffer,Timer_pacer,Timer_1,Rel
ay_d1_s7,Relay_d1_s8,Relay_d1_s9,Relay_d1_s10
10808 COM /Names/ A_d3,A_d4,D_a1,D_a2,D_a3,D_a4
10809 COM /Cal_fact/ Cal_fact(20,20),Avg_volts(20,20),J_chan_pt,J_chan_beg,Dvm_v
olts(100),Ad_volts(110),Load_gain(130),Ircnt,Xoff_fit,Yoff_fit
10810 !
```

```
10811 Chan=7
10812 Ps_val=9
10813 WAIT 2.97
10814 FOR I=2 TO 21
10815 Vstep=.9
10816 IF (I=12) THEN
10817 !
10818 ! REVERSE POLARITY OF PS18
10819 !
10820    CALL Set_rela_d1_s10(1,1)
10821 !
10822 END IF
10823 IF (I>=12) THEN Vstep=-.9
10824 Ps_val=Ps_val-Vstep
10825 Ps_val$=VAL$(Ps_val)
10826 Stop(Scanner)
10827 Ftic_setup("SET_PS18_VOLTAG",Ps_val$)
10828 WAIT 2
10829 CALL Read_dvm(Dvm_no,Chan,Units$,Reading)
10830 !WAIT 2
10831 !CALL Read_dvm(Dvm_no,1,Units$,Readb)
10832 WAIT 2
10833 !Reading=Reading-Readb
10834  IF (I=11) THEN
10835  PRINT "REVERSE POLARITY"
10836  INPUT Anss$
10837  END IF
10838 Start(Scanner)
10839 Dvm_volts(I)=Reading
10840 IF (I<>21) THEN
10841 WAIT 2.99
10842 END IF
10843 NEXT I
10844 SUBEND ! DRV_DA_RD_DVM
10845 !
10846 !     ***********************************************************
10847 !
10848 SUB Set_db_out(Atten_grp,Dba,Dbb,Test_no)
10849 !
10850 COM /Cal_settings/ Settings(6)
10851 !
10853 !  THIS ROUTINE DETERMINS THE GAUUSSIAN ATTENUATOR SETTINGS.
10855 !
10856 !
10857 ALLOCATE Db_ray(2)
10858 ALLOCATE Vin(2)
10859 ALLOCATE Atten$(2,2)[10]
10860 ALLOCATE Db_setting(2)
10861 Atten$(1,1)="ATTEN_A"
10862 Atten$(1,2)="ATTEN_B"
10863 Atten$(2,1)="ATTEN_C"
10864 Atten$(2,2)="ATTEN_D"
10865 CALL Init_pg_at
10866 Ftic_setup("CAL_GEN1")
10867 Ftic_setup("CAL_GEN2")
10868 Db_ray(1)=Dba
10869 Db_ray(2)=Dbb
10870 Spect_vin=10
10871 Vinput=5
10872 Chan_in=0
10873 IF (Atten_grp=2) THEN Chan_in=3
10874 FOR Jkl=1 TO 2
10875 Chan_out=Chan_in+Jkl
10876 Spect_db=Db_ray(Jkl)
```

```
10877 !
10878 !  FIND THE VALUE OF THE GAUSSIAN INPUT IMPLIED BY THE SPECIFICATION.
10879 !
10881 Vout=Spect_vin/(10^(Spect_db/20))  ! SPECT. VOUT
10882 PRINT "VOUT=",Vout
10883 !
10884 !  DETERMINE A NEW VALUE FOR db BASED ON A 5 VOLT ATTENUATOR INPUT.
10885 !
10887 Db=20*LGT(Vinput/Vout)                  ! DB BASED ON A 5V INPUT
10888 !
10889 !  SET THE ATTENUATORS AND THE PULSE GENERATOR TO DELIVER VOUT.
10890 !
10891 !   IF (Db>50) THEN Db=50     ! DBs OVER 50 MAY REQUIRE INPUTS GREATER
10892                             ! THAN THOSE ALLOWED BY THE ATTENUATOR.
10893 Re_do: !
10894 Vin$=VAL$(Vinput)
10895 IF (VAL(Vin$)>7) THEN STOP        ! THE NEED FOR THIS KIND OF CHECK IS
10896                                   ! HIGHLY DEPENDENT ON INPUT AND OUTPUT
10897                                   ! LOADING.  THE DB SETTING IS ALSO A
10898                                   ! LOADING FACTOR.
10899 CALL Set_pulse_gen(Atten_grp,Vin$,Settings(*))
10900 PRINT "LEAVING SET_PULSE_GEN"
10901! Ftic_setup("INIT_SCOPE")
10902 Db_send=DROUND(Db,2)
10903 Db_setting(Jkl)=Db_send
10904 CALL Set_atten(Atten$(Atten_grp,Jkl),Db_send,Db_true,"NEW")
10905 PRINT "LEAVING SET_ATTEN"
10906 !
10907 !  MEASURE THE db REALIZED BY THE SETTINGS.
10908 !
10910 CALL True_db_dvm(Atten_grp,Chan_in,Chan_out,Db_test,Vin_read,Vout_read)
10911 !
10912 !  MAKE ADJUSTMENTS TO SUPPLIES AND ATTENUATORS IF NECESSARY.
10913 !
10915 !  Vin(Jkl)=Vout*10^(Db_true/20)
10916    Vin(Jkl)=Vout*10^(Db_test/20)  ! USE DB_TEST SO THAT THE ACTUAL
10917                                   ! DB IS USED IN THIS CALCULATION.
10918                                   !
10919 IF (Vin(Jkl)>7) THEN              ! THE NEED FOR THIS KIND OF CHECK IS
10920                                   ! HIGHLY DEPENDENT ON INPUT AND OUTPUT
10921                                   ! LOADING.  THE DB SETTING IS ALSO A
10922                                   ! LOADING FACTOR.
10923 PRINT "VIN WAS GREATER THAN 7 VOLTS."
10924 INPUT Ans$
10925 Db=Db-5
10926 GOTO Re_do
10927 END IF
10928 ! Vin$=VAL$(Vin(Jkl))
10929 ! CALL Set_pulse_gen(Vin$)
10930 NEXT Jkl
10931 !
10932 ! NOW DETERMINE WHICH ATTENUATOR MUST BE MODIFIED.
10933 !
10934 IF (Vin(2)>Vin(1)) THEN
10935 END IF
10936 Klm=2
10937 Mkl=1
10938 IF (Vin(1)>Vin(2)) THEN
10939 Klm=1
10940 Mkl=2
10941 END IF
10942 Db_dif=20*LGT(Vin(Klm)/Vin(Mkl))
10943 !   Db_dif=1               !**********************
10944  !
```

```
10945  !  THE FOLLOWING SAVES ATTENUATOR VALUES BEFORE ADJUSTMENT.
10946  !
10947   IF (Atten_grp=1) THEN
10948   Settings(3)=INT(Db_setting(1)/10)*10+1  !ADD 1 TO REFLECT THE ONE
10949   Settings(4)=INT(Db_setting(2)/10)*10+1  !DB ADDED IN SUB SET_ATTEN
10950   ELSE
10951   Settings(5)=INT(Db_setting(1)/10)*10+1
10952   Settings(6)=INT(Db_setting(2)/10)*10+1
10953   END IF
10954 !
10955 IF (Db_dif>.5) THEN
10956! Db_send=DROUND(Db_dif,1)
10957 Db_send=INT(Db_dif+.5)
10958 CALL Set_atten(Atten$(Atten_grp,Mkl),Db_send,Db_true,"NEW")
10959  IF (Db_true<(2*Db_dif)) THEN
10960  PRINT "DB_TRUE=",Db_true
10961  !
10962  !  THE FOLLOWING SAVES THE ADJUSTED ATTENUATOR VALUE.
10963  !
10964  Indx=Mkl+2
10965  IF (Atten_grp=2) THEN Indx=Indx+2
10966  Settings(Indx)=INT(Db_setting(Mkl)/10)*10+Db_send+1 !ADD 1 (ABOVE REASON)
10967  !
10968  !  THE FOLLOWING IS ONLY NEEDED WHEN PGENs AND ATTENs WILL NOT
10969  !  BE SET FROM AN ATTENUATOR SETTINGS FILE.
10970  !
10971  GOTO Here
10972  PRINT "SETTING ATTENUATOR TO TENS PART OF DB."
10973  CALL Set_atten(Atten$(Atten_grp,Mkl),Db_setting(Mkl),Db_true,"NEW")
10974  PRINT "SETTING ATTENUATOR TO ONES PART OF DB."
10975  Ftic_setup(Atten$(Atten_grp,Mkl),"X"&VAL$(Db_send))
10976  ELSE
10977  PRINT "SETTING ATTENUATOR TO TENS PART OF DB."
10978  Db_trunc=INT(Db_setting(Mkl)/10)*10
10979  Ftic_setup(Atten$(Atten_grp,Mkl),"Y"&VAL$(Db_trunc))
10980!  CALL Set_atten(Atten$(Atten_grp,Mkl),Db_setting(Mkl),Db_true,"NEW")
10981 Here:!
10982  END IF
10983 END IF
10984 Vin$=VAL$(Vin(Klm))
10985 CALL Set_pulse_gen(Atten_grp,Vin$,Settings(*))  ! SET THE PULSE GEN. TO A
VIN THAT WILL
10986                          ! PRODUCE VOUT USING THE TRUE DB SETTING.
10987! CALL True_db_dvm(Atten_grp,Chan_in,Chan_out,Db_true,Vinfin,Voutfin)
10988! PRINT "DB=",Db_true,"VIN=",Vinfin,"VOUT=",Voutfin
10989! PRINT "LEAVING SET_DB_OUT"
10990 !
10991 !  STORE ATTENUATOR SETTINGS
10992 !
10994 CALL Write_atten_dat(Test_no,Settings(*))
10995 !
10997 SUBEND !SET_DB_OUT
10998 !
10999 !  ****************************************************************
11000 !
11002 SUB Burst_setup(Trigs1,Idle_time_1)
11003 !
11004 !  THIS ROUTINE IS USED TO SET UP CARDS FOR SAMPLING DATA IN BURSTS.
11005 !
11006 COM /Names/ Scanner,Cnt_tot,A_d1,A_d2,Multi_buffer,Timer_pacer,Timer_1,Rel
ay_d1_s7,Relay_d1_s8,Relay_d1_s9,Relay_d1_s10
11007 COM /Names/ A_d3,A_d4,D_a1,D_a2,D_a3,D_a4
11008 !
11009 !  CLOSE MASTER RESET UNTIL RELAY AND CARD SETTINGS ARE COMPLETE.
```

```
11010 !
11011    CALL Set_rela_d1_s10(16,1)
11012 !
11013 !  OPEN "TIMER PACER COP-NOT" TO "SCANNER_EXT" RELAY.
11014 !
11015 !  CALL Set_rela_d1_s7(10,0)
11016 !
11017 !  ENERGIZE THE DECODER TO BE USED FOR THE FOLLOWING RELAY CONDITIONS
11018 !  AND CAUSE THE FOLLOWING TO HAPPEN:
11019 !  CLOSE "TIMER PACER EXT-NOT" TO "COUNTER BORROW-NOT" RELAY.
11020 !  OPEN "TIMER PACER EXT_NOT" TO "COUNTER CARRY-NOT" RELAY.
11021 !  SWITCH IN CARRY & BORROW "AND" GATE.
11022 !  CLOSE COUNTER INP_1.
11023 !
11024    CALL Set_rela_d1_s7(14,1)
11025 !
11026 !  RESET TIMER_PACER & CNT_TOT EOPS IN CASE OF PREMATURE TRIGGERS.
11027 !
11028    WAIT .5
11029    Check_done(Timer_pacer,Dum)
11030    WAIT 5
11031    Check_done(Cnt_tot,Dum)
11032    WAIT 5
11033    Preset(Timer_pacer,3.1)       ! SETS TIMER_PACER TO A 3.1 SEC. DELAY
11034 !  LOCK SCAN CONTROL PACER'S EXTERNAL TRIGGER
11035    Disable_extrig(Scanner)
11036    WAIT 2.0E-2
11037 !  CLOSE "SCANNER EXT-NOT" TO "BURST INITIATION SIGNAL" RELAY.
11038    CALL Set_rela_d1_s7(12,1)
11039    WAIT 2.0E-2
11040    CALL Set_rela_d1_s10(16,0)    !OPEN MASTER RESET
11041    WAIT 2.0E-2
11042 !
11043 !  CLOSE FUNCTION GEN., TIMER_PACER EOP AND, CNT_TOT EOP TO D_A4, LINES
11044   CALL Set_rela_d1_s8(4,0)
11045 !
11046 !  LOAD COUNTER WITH DELAY COUNT.
11047 !
11048    Trigs=637   ! 637 IS THE NUMBER OF 7.7ms COUNTS TO DELAY 4.9 SEC.
11049    Output(Cnt_tot,Trigs)
11050 !
11051 !  LOAD COUNTER TOTALIZER'S SECOND RANK STORAGE WITH A BURST COUNT.
11052 !
11053    Trigs=65535-39
11054    Preset(Cnt_tot,Trigs)
11055 !
11056 !  DISABLE A_D2
11057 !
11058    Disable_mxctrl(A_d2)
11059    Disable_extrig(A_d2)
11060 !
11061 !  SET FUNCTION GENERATOR
11062 !
11063    Ftic_setup("SET_FREQ_GEN1","FU2FR260HZAM2VOOF1.01VO")
11064 !
11065 Disable_extrig(A_d1)
11066 !  START TEST
11067 !
11068 PRINT "TYPE ENTER TO START 15 SEC TEST"
11069 INPUT Ans$
11070 CALL Set_rela_d1_s7(9,1)
11071 Wait_for(Cnt_tot)
11072 Enable_extrig(A_d1)
11073 WAIT 2.0E-2
```

```
11074 CALL Set_rela_d1_s7(9,0)
11075     WAIT 10.1    ! WAIT FOR FIRST PART OF FOUR PART TEST TO FINISH
11076     Preset(D_a4,0)
11077     Start(D_a4)
11078     CALL Set_rela_d1_s10(5,1)
11079     CALL Set_rela_d1_s7(15,1)
11080     CALL Set_rela_d1_s7(16,1)
11081     WAIT .5                        ! WAIT FOR SECOND PART TO FINISH
11082 !
11083 !   LOCKOUT MEMORY
11084 !
11085     Enable_lockout(Multi_buffer)
11086     WAIT 2.0E-2
11087 !   CALL Set_rela_d1_s10(16,1)
11088 !   WAIT 2.0E-2
11089     Stop(Scanner)
11090     Ftic_setup("SET_PS1B_VOLTAG","0")
11091 !   CALL Set_rela_d1_s9(2,0)
11092 !   CALL Set_rela_d1_s7(15,0)
11093 !   CALL Set_rela_d1_s7(16,0)
11094 !   WAIT 2.0E-2
11095 !   Ftic_setup("SET_PS1B_VOLTAG","5")
11096 !
11097 !   CLOSE "TIMER EXT-NOT" TO "COUNTER CARRY-NOT RELAY
11098 !   OPEN "TIMER EXT-NOT" TO "COUNTER BORROW-NOT" RELAY
11099 !   OPEN INP_1
11100 !   OPEN COUNTER AND GATE
11101 !
11102     CALL Set_rela_d1_s7(14,0)
11103     CALL Set_rela_d1_s10(4,1)
11104     WAIT 2.0E-2
11105 !
11106 !   CLOSE "TIMER PACER COP-NOT" TO "SCANNER_EXT" RELAY.
11107 !
11108 !   CALL Set_rela_d1_s7(10,1)
11109 !   WAIT 2.0E-2
11110 !
11111 !   RE-SET COUNTER AND TIMER/PACER
11112 !
11113     Cnts=65535-Trigs1
11114     Output(Cnt_tot,Cnts)
11115     Preset(Timer_pacer,Idle_time_1)
11116 SUBEND  ! BURST_SETUP
11117 !
11118 !   *******************************************************************
11119 !
11121     SUB Store_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
11122     DIM Temp_aray(4000)
11123     ASSIGN @File TO "/FTM/HOME_DIR00/RUN_DATA:,1400"
11124     ASSIGN @Param TO "/FTM/HOME_DIR00/RUN_DATA_EX:,1400"
11125     OUTPUT @Param;Tot_samples,Wait_time
11126     !
11127     !  LOAD UP TEMP ARRAY AND STORE DATA 4000 PER RECORD
11128     !
11129     FOR J=1 TO 5
11130     Low=(J-1)*4000
11131     High=(J*4000)-1
11132     Kt=0
11133     FOR I=Low TO High
11134     Kt=Kt+1
11135     Temp_aray(Kt)=Volts(I,0)
11136     NEXT I
11137     OUTPUT @File,Rec_num;Temp_aray(*)
11138     Rec_num=Rec_num+1
```

```
11139   NEXT J
11140   SUBEND !STORE_RUN_DATA
11141   !
11142   !  *********************************************************************
11143   !
11145   SUB Get_run_data(Volts(*),Tot_samples,Wait_time,Rec_num)
11146   DIM Temp_aray(4000)
11147   ASSIGN @File TO "/FTM/HOME_DIR00/RUN_DATA:,1400"
11148   ASSIGN @Param TO "/FTM/HOME_DIR00/RUN_DATA_EX:,1400"
11149   ENTER @Param;Tot_samples,Wait_time
11150   !
11151   !  GET DATA 4000 AT A TIME AND STORE IN VOLTS ARRAY
11152   !
11153   FOR J=1 TO 5
11154   ENTER @File,Rec_num;Temp_aray(*)
11155   Low=(J-1)*4000
11156   High=(J*4000)-1
11157   Kt=0
11158   FOR I=Low TO High
11159   Kt=Kt+1
11160   Volts(I,0)=Temp_aray(Kt)
11161   NEXT I
11162   Rec_num=Rec_num+1
11163   NEXT J
11164   SUBEND !GET_RUN_DATA
11165   !
11170 SUB Squib_driv(Analyze)      !  PAR 68
11171 COM /Test_config/ Time_per_chan_1,Time_per_chan_2,Time_per_chan_3,Window_1
,Window_2,Window_3,Name$(20)[10],W1l1,W1l2,W2l1,W2l2,W2l3,W2l4,W3l1,W3l2
11172 COM /Test_config/ Chans_per_scn_1,Chans_per_scn_2,Chans_per_scn_3,Idle_tim
e_1,Idle_time_2
11174 COM /Test_select/ Fil$(1:72)[80],Mark$[2],Test_name$[80]
11175 COM /Switch/ Sw_state(4),Srh,Sw_windows,Wait_gate(3)
11176    IF (Fil$(12)[3;2]="" ") THEN
11177 COM /Anal_data/ Line_name$(20)[10],Voltage_nom$(32)[10],Voltage_tol$(32)[1
0],Time_edge_nom$(16)[10],Time_edge_tol$(16)[10],Time_ref,Time_edge_ignor(16)
11178 COM /Anal_data/ Time_ref_old
11179 DATA "0","0","0","0","0","0","0","0","0"
11180 DATA     "10.77","10.77","10.8","22.6","24.5","22.6","25.3","25.3"
11181 DATA     "0","0","0","0","0","0","0","0"
11182 DATA     "0","0","0","0","0","0","0","0"
11183 READ Voltage_nom$(*)
11184 DATA "0",".4",".4",".4",".4",".4",".4",".4",".4"
11185 DATA     ".005",".005",".005",".005",".005",".005",".005",".005"
11186 DATA     ".4",".4",".4",".4",".4",".4",".4",".4"
11187 DATA     ".4",".4",".4",".4",".4",".4",".4",".4"
11188 READ Voltage_tol$(*)
11189 DATA "0",".005",".005","0.21","1","1","1","1.9","0.11"
11190 DATA     ".025",".025",".23","1.02","1.02","1.02","1.92",".13"
11191 READ Time_edge_nom$(*)
11192 DATA "0",".005",".005","0.1","0.2","0.2","0.2","0.2","0.05"
11193 DATA     ".005",".005","0.1",".2",".2",".2",".2","0.05"
11194 READ Time_edge_tol$(*)
11195 DATA 0,0,0,0,0,0,0,0,0
11196 DATA   0,0,0,0,0,0,0,0
11197 READ Time_edge_ignor(*)
11198 !
11204 PRINTER IS 701
11205 Name$(0)="EA1_SQ_DR"
11206 Name$(1)="EA2_SQ_DR"
11207 Name$(2)="30V BAT"
11208 Name$(3)="SGG SQ DR"
11209 Name$(4)="GAS SQ DR"
11210 Name$(5)="SGS SQ DR"
```

```
11211 Name$(6)="WUN SQ DR"
11212 Name$(7)="WEX SQ DR"
11213 !
11214 FOR I=1 TO 20
11215 K=I-1
11216 Line_name$(I)=Name$(K)
11217 NEXT I
11218 CALL Spec_check
11219 STOP
11221 !
11222 Test_name$=Fil$(12)[5]
11223 !  SPECIFY THE Srh SWITCH USED TO START THE TEST.
11224 !  7 - EP & GUIDANCE       8 - SEEKER
11225 !
11226   Srh=7
11227 !
11228                   ! ONLY BECAUSE IT ALSO DISABLES THE ATTENUATOR SWITCH
11229                   ! USED IN THE Test_frame SUBROUTINE.
11230                   !
11231 !
11232   Time_ref=0
11233 !
11234 FOR Ijk=1 TO 8   ! ANALYZE ALL 8 DATA LINES
11235    SELECT Ijk
11236     CASE 1,2
11237     Time_ref=0
11238     CASE ELSE
11239     Time_ref=Time_ref_old
11240    END SELECT
11241 !
11242 !
11243 !
11244 !
11245 ! ESTABLISH WINDOW PARAMETERS.
11246 !
11247 Time_per_chan_1=1.E-4
11248 Time_per_chan_2=2.E-3
11249 Time_per_chan_3=5.E-4
11250 Window_1=3.5E-2
11251 Window_2=3.5
11252 Window_3=2.5
11253 Idle_time_1=5.0E-2
11254 Idle_time_2=1.00E-1
11255 !
11256 !  SET SCANNER CHANNEL POINTERS ACCORDING TO (SCANNER#-1)*32+CHANNEL#
11257 !
11258 W1l1=0
11259 W1l2=32
11260 !
11261 W2l1=1
11262 W2l2=33
11263 W2l3=65
11264 W2l4=97
11265 !
11266 W3l1=2
11267 W3l2=34
11268 !
11269 !  IDENTIFY CHANNEL ASSIGNMENTS IN MEANINGFUL TERMS.
11270 !
11271 !
11272 !  ANALYZE DATA
11273 !
11274 Ikk=Ijk-1
11275 CALL Test_frame("P",VAL$(Ikk))     ! GETS DATA AND PUTS IT INTO VOLTS()
```

```
11276                                     ! ARAYS
11277 CALL Main_frame(Ikk,Test_name$)
11278 !
11279 NEXT Ijk
11280    END IF
11281 Ftte_skip_test
11282 SUBEND
11283 !
11284 !     ******************************************************
11285 !
```

DISTRIBUTION LIST

Commander
Armament Research, Development and Engineering Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCAR-IMI-I (5)
SMCAR-FSP-S (10)
Picatinny Arsenal, NJ 07806-5000

Commander
U.S. Army Armament, Munitions and Chemical Command
ATTN: AMSMC-GCL (D)
Picatinny Arsenal, NJ 07806-5000

Administrator
Defense Technical Information Center
ATTN: Accessions Division (12)
Cameron Station
Alexandria, VA 22304-6145

Director
U.S. Army Materiel Systems Analysis Activity
ATTN: AMXSY-MP
Aberdeen Proving Ground, MD 21005-5066

Commander
Chemical Research, Development and Engineering Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCCR-MSI
Aberdeen Proving Ground, MD 21010-5423

Commander
Chemical Research, Development and Engineering Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCCR-RSP-A
Aberdeen Proving Ground, MD 21010-5423

Director
Ballistic Research Laboratory
ATTN: AMXBR-OD-ST
Aberdeen Proving Ground, MD 21005-5066

Chief
Benet Weapons Laboratory, CCAC
Armament Research, Development and Engineering Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCAR-CCB-TL
Watervliet, NY 12189-5000

Commander
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCAR-ESP-L
Rock Island, IL 61299-6000

Director
U.S. Army TRADOC Systems Analysis Activity
ATTN: ATAA-SL
White Sands Missile Range, NM 88002